



Faculty of Computer Science & Information Technology
University of Malaya



Perpustakaan SKTM

“INTERACTIVE 3D TOUR AROUND FSKTM”

Prepared by,

KHAIRUL EFEEZA BIN ISMAIL
WET 990222
2003/2004

Supervisor : EN. Amirrudin Hj. Kamsin

Moderator : Pn. Nornazlita Hussin

Bachelor of Information Technology,

Faculty of Computer Science and information Technology,

University of Malaya

ACKNOWLEDGEMENTS

I would like to gratefully acknowledge the contributions of several people who have helped me to complete this project.

First, I would like to convey my grateful thanks to my supervisor, En Amirudin Hj. Kasmin for giving me valuable assistance to overcome most of my problems in my work. He has given me a lot of advices, comments and guidelines through out the completion of this project.

Second, I would like to thank to my moderator, Pn Nornazlita Hussin, for giving me a lot of valuables suggestions and comments.

Third, I would like to express my appreciation to my father and sister whom gave me encouragement, support and advice. Especially I would like to thank my sister, Khainur Aleeza Ismail. She has helped me a lot to overcome several problems that I face during the implementation of this project.

I would like to tribute this report to my mother, Allahyarhamah Khadijah Jali, as she always want me to achieve better everyday. Al-Fatihah.

Last but not least, I would like to thank to ALLAH for giving me strength, spiritual motivation and guidance to successfully complete this report also all those who helped me that I not mention here. Thank you very much.

ABSTRACT

Now days, 3D's designs has become popular among the top designers. Even now revolutionaries of the cell phone screen interface had become in 3D's. 3D's also had been used worldwide as a virtual mapping of an interest place as part of their information systems. According to this thesis title – Interactive 3D Tour Around FSKTM is an interactive 3D virtual FSKTM system. This system enables the users to view FSKTM virtually easily and quickly via computers. By using this system, the user can feel and see virtually FSKTM environments without have to come physically to FSKTM. Interactive environments will power this system as part of the interactive way of making people attractively using this system. Target of the system is to providing FSKTM with 3D's information technologies as 3D's design are something new here and with this system will ease FSKTM visitors or new students to find a certain place in FSKTM. The target users are students it self, lecturer as well as tourist. Using 3DGameStudio as well as 3DstudioMax as the software development tools will develop this system. The system is developed on Microsoft Window XP operating system due to its stability. With this system, hopefully 3D map will be using worldwide as part of their informational. Furthermore, hope the faculty (FSKTM) will use this project as it informational of the faculty as the first ever design of virtual FSKTM faculty.

Project Aim

This documentation outlines the research, design, development and evaluation of the production of an interactive 3D model that can be used as an aid for visitors to the FSKTM. At the initial planning phase of the project, a set of minimum requirements and objectives were decided upon. These goals were to be met during the development of the project. The minimum requirements were:

- To investigate the most appropriate method of producing 3D rendered model of the FSKTM.
- To learn the key features of the chosen 3D modeling language.
- To produce a sample 3D area using the chosen method for evaluation purposes.
- To evaluate software tools that could assist in the creation of the 3D area.

The objectives of the project are:

- To produce a 3D model of the FSKTM campus to be used as an interactive map.
- To evaluate the usability of the 3D model as a map.
- To use texture mapping or another appropriate method to create a more realistic 3D environment.
- To obtain opinions from testing to assess suggested improvements to the environment.

TABLE OF CONTENTS

TABLE OF CONTENTS

1.0	ABSTRACT.....	i
2.0	ACKNOWLEDGEMENT.....	ii
3.0	PROJECT	
	AIM.....	iii
4.0	TABLE OF CONTENTS.....	iv
5.0	LIST OF FIGURES.....	xi
6.0	CHAPTER 1 : INTRODUCTION.....	1
6.1	Interactive 3D Tour Around FSKTM.....	1
6.2	Computer Graphic.....	1
6.2.1	History of Computer Graphic.....	2
6.3	Three Dimensional (3D)	4
6.3.1	History and Method For 3D Generation.....	4
6.3.1.1	Using Hardware.....	4
6.3.1.2	Software Solution.....	5
6.4	What Makes a Picture 3D	6
6.5	What Are 3D Graphic.....	7
6.6	Creating a Virtual 3D World.....	8
6.7	What Part of the Virtual World Shows on the Screen.....	8
6.8	How to Make it Look Real.....	9
6.8.1	Shapes.....	9
6.8.2	Surface Texture.....	10

6.8.3	Lighting.....	11
6.8.4	Perspective.....	11
6.8.5	Depth of Field.....	12
6.8.6	Anti-Alias.....	12
6.9	System Overview.....	13
6.9.1	System Objective.....	13
6.9.2	System Scope.....	13
6.9.3	System Requirements.....	13
6.10	Introduction of Project – Interactive 3D Tour Around FSKTM.....	15
7.0	CHAPTER 2 : LITERATURE REVIEW.....	17
7.1	Purpose.....	17
7.2	The Study On Current Software Development Tools.....	17
7.2.1	OpenGL.....	18
	7.2.1.1 Advantages.....	19
	7.2.1.2 Disadvantages.....	19
7.2.2	VRML.....	19
	7.2.2.1 Advantages.....	20
	7.2.2.2 Disadvantages.....	20
7.2.3	Java3D.....	21
	7.2.3.1 Advantages.....	21
	7.2.3.2 Disadvantages.....	22
7.2.4	Clickable Image Maps.....	22

7.2.4.1 Advantages.....	23
7.2.4.2 Disadvantages.....	23
7.2.5 3DGameStudio.....	24
5.2.3.2 Advantages.....	24
5.2.3.3 Disadvantages.....	25
5.3 The Study On Other Existing System.....	25
5.3.1 Virtual Los Angeles.....	25
5.3.2 The 3D Internet Site Netrovia.....	27
5.3.3 WWW.STONYPLAIN.NET.....	28
8.0 CHAPTER 3 : METHODOLOGY.....	30
8.1 Building 3D Worlds.....	30
8.2 Recognition of Problems.....	30
8.3 Telepresence.....	31
8.4 Primary Sources.....	32
8.5 Secondary Sources.....	33
8.6 Project Scale.....	33
8.7 Questionnaires.....	33
8.8 Project Development Life Cycle.....	34
8.8.1 Software Process Model.....	34
8.9 Justification of Propose Methodology.....	38
9.0 CHAPTER 4 : SYSTEM ANALYS.....	41

9.1	Requirements Analysis.....	41
7.1.1	Functional Requirements.....	41
7.1.2	Non-Functional Requirements.....	42
7.1.2.1	Reliability.....	42
7.1.2.2	Efficiency.....	42
7.1.2.3	User Friendliness.....	42
7.2	Hardware Requirements.....	43
7.3	Software Requirements.....	43
7.3.1	Operating Systems (OS) Software.....	43
7.3.2	Software Development Tools : 3DGameStudio.....	44
7.3.2.1	The Editors.....	44
7.3.2.2	The Engines.....	45
7.3.2.3	Model Editor (MED – model editor design)	46
7.3.2.4	World Editor Design (WED)	49
7.3.2.4.1	Map Editing Concepts.....	50
7.3.2.4.2	Object Oriented Concepts.....	50
7.3.2.5	Rendering 3D Design.....	52
7.4	Techniques Used to defined Requirements.....	53
10.0	CHAPTER 5 : SYSTEM DESIGN.....	57
10.1	Overview.....	57
10.2	System Architecture Design.....	58
10.3	System Functionality Design.....	58

10.3.1	System Structure Charts.....	59
10.3.2	Data Flow Diagram (DFD)	59
10.4	User Interface Design.....	61
10.5	Design Process.....	61
11.0	CHAPTER 6 : SYSTEM TESTING.....	63
11.1	Introduction.....	63
11.2	Testing Strategies.....	63
11.3	System Testing.....	64
11.3.1	Unit Testing.....	65
11.3.2	Integration Testing.....	65
11.4	Conclusion.....	65
12.0	CHAPTER 7 : SYSTEM EVALUATION.....	67
12.1	Problems Encountered and Solutions.....	67
12.1.1	Difficulties In Determining The Scope Of The System.....	67
12.1.2	Difficulty In Designing Telepresence Environments.....	67
12.1.3	Difficulty In Rendering The System.....	68
12.2	System Strengths.....	68
12.2.1	User Friendliness.....	68
12.2.2	Easy To Use.....	68
12.2.3	Easy Accessible.....	68
12.2.4	Real-Time Rendering.....	69

12.2.5	Free From 3 rd Party Platform.....	69
12.3	System Limitations.....	69
12.3.1	Lack Of Detail Structure.....	69
12.3.2	Limited Functionality.....	70
12.4	Future Enhancement.....	70
12.4.1	Enhance User Interface.....	70
12.4.2	More Functionality Added.....	70
12.4.3	Develop For Other Platform.....	71
12.4.4	Provide In Other Language.....	71
13.0	CHAPTER 8 : DISCUSSION	73
13.1	Modeling and Animation.....	73
13.1.1	Modeling.....	73
13.2	Problem Encountered.....	75
13.3	Suggestion.....	76
14.0	CONCLUSION	78
	APPENDIX A: USER MANUAL	80
	APPENDIX B: QUESTIONNAIRE	87
	APPENDIX C: WDLMAN (SCRIPT)	90
	APPENDIX D: PROJECT SCEDULE	110
	REFERENCE	112

LIST OF FIGURES

List Of Figures

Figure 1.0 Overlaying the images for the left and the right eye into an Autostereogram.....6

Figure 1.1 Triangles.....7

Figure 2.0 Clickable Image Maps.....22

Figure 2.1 Virtual Los Angeles.....26

Figure 2.2 Virtual shopping.....27

Figure 2.3 Youth Bike Park.....28

Figure 3.0 The Waterfall Model with Prototyping.....35

Figure 4.0 Work Flow of 3DGameStudio.....52

Figure 5.0 System Architecture Of The System.....58

Figure 5.1 Structure Chart.....59

Figure 5.2 Gane and Sarson Notation of Data Flow Diagram.....60

Figure 8.1 Models For Represent User.....73

Figure 8.2 Models Being Added To World Design.....74

Figure 8.3 Exporting Worlds To Complete Map.....74

Figure 8.4 Problems Encountered.....75

CHAPTER 1

INTRODUCTION

Chapter 1: INTRODUCTION

1.0 Interactive 3D Tour Around FSKTM

This chapter will be a quick review of the project (Interactive 3D Tour Around FSKTM), computer graphics as well as 3D history.

1.1 COMPUTER GRAPHICS

Computer Graphics is concerned with theories and techniques to input, output, generate, transform, manipulate and transmit pictures (or, more generally, visually meaningful data) with the aid of computers; its objects are artifacts obtained by synthesis [1]. In contrast, picture or image processing deals with the images obtained from the real world; it is a field, quite distinct from computer graphics although some overlap exists [1]. Graphics are a natural and efficient way of communicating information. Computer graphics is a branch of computer science. Computer graphics is applied in Computer Aided Design (CAD) presentation graphic, computer art (fine art and commercial art), entertainment (motion picture, music video, and television show) education and training (simulator), visualization (scientific and business visualization) image processing and graphical user interface [6]. Computer graphics is also used in guiding satellites, rockets and space shuttles [5]. The topics of computer graphics include output primitive, 2D transformations, 2D viewing, 3D object representation, 3D transformations, 3D viewing, visible surface detection and surface rendering. 3D graphics is a graphic system that a 3D data set is converted to 2D viewing.

1.1.1 HISTORY OF COMPUTER GRAPHICS

In the middle 50's and to the early 60's was the beginning era. The early 60's to the late 60's was the "gee whiz, look what aerospace automotive is doing" era and in late 60's to the early 70's was the "let's form a new graphic economy" era [1].

1950

In 1950, the first computer driven display, attached to Massachusetts Institute of technology's (MIT) Whirlwind 1 computer, was used to generate simple pictures [7]. This display made use of a cathode-ray tube (CRT) [4]. During 1950's, interactive computer graphics made little progress because the computers of that period were so unsuited to interactive use. These computers were "number crunches". Only toward the end of 1950's, with the development of machines like MIT's TX-0 and TX-2 did interactive computing become feasible and interest in Computer Graphics then began to increase rapidly [4]. In 1958, Cal comp 565 drum plotter was introduced [1].

1960

In the spring of 1963, Dr Ivan Sutherland at MIT described sketchpad with some of the seminal data-structure work laying the software theoretical basis for Computer Graphics. The system uses a light pen to draw pictures on the display [1]. This promoted interactive Computer Graphics. Around 1963, Steve Coons began developing surface-patch (known as texture-mapping right know) techniques for computer graphics modeling [1]. In the early 1960's, there were beam-foaming

techniques, dot raster system and line raster systems [1]. By mid 1960's, large computer graphics research projects underway at MIT, General Motors, Bell Telephone Laboratories, and Lockheed Aircraft where the Golden Age of computer graphics began [4]. The ACM special Interest Committee for Computer Graphics was formed in late 1966 and became a Special Interest Group-SIGGRAPH in 1969 [1]. The applications in 1960's included computer-aided design in the aircraft and textile industries, management information systems simulations, process control, computer-aided education, pattern recognition, graphic arts and computer generated movies [1].

1970

The early system use keyboard and light pens. Graphic tablets, digitizers and touch-sensitive devices were included later [1]. Early systems require the user to create their own software. In 1960's a wide variety of propriety package were available. Since early 1970's, complete turnkey systems have become available [1]. Throughout 1960's and early 1970's computer graphics devices were considered "expensive toys" [1]. In late 1960's and early 1970's a number of new computer graphics companies were organized [1]. In the early 1970's the development of semiconductors and integrated memory hardware lower the size and cost pixel image storage systems, and raster devices were available commercially [2].

1.2 THREE-DIMENSIONAL (3D)

3D visualization tools are indispensable and have numerous uses. Providing a 3D image enhances the first impression of a design and impresses anyone. The color and texture of almost any material can be simulated and incorporated into images, previewed in its natural environment by superimposing the 3D image into almost any photograph. Using 3D animation instead of a real-life video shoot has several advantages, including a high level of control over the way the model appears. This can be accomplished by using different lighting setups in various settings. Camera shots that are impossible in the real world, can be created on the computer.

1.2.1 HISTORY AND METHOD FOR 3D-GENERATION

1.2.1.1 Using Hardware:

The first method for producing three-dimensional-impression was the Wheatstone Stereoscope (named by the British physician Charles Wheatstone who did research on the Stereo-Disparity in 1828 and invented the Stereoscope in 1932) [10]. It consists of two L-shaped viewing tubes that feed a pair of stereo-images to the viewer's right and left eyes. Those Stereoscopes are used in science (for three-dimensional-visualization) but also for amusement (e.g. in night-clubs). In the Sixties, the German Matell-Company sold millions of their "View master"-Stereoscope.

Tachistoscopic devices Shutterglasses [10] - Tachistoscopic devices use time-division-multiplexing for alternately displaying the image for the right and the left eye. Some VR-Systems use this method, but instead of shutter glasses and a mechanical picture-projector they usually make use of LCD-displays.

Anaglyphs Polarization and Red-Green-Glasses [10] - Anaglyphs where the first approach to combine the image for the right and the left eye into one picture. The clue is that both images are drawn with different color or different polarization. When the viewer uses special glasses, like Polarization-Glasses or Red-Green-Glasses, each of his eyes is presented only one image (out of two) and his brain is able to reconstruct the three-dimensional-impression. This method was tested in TV and cinema, but did not really succeed.

1.2.1.2 Software solution:

The new approach: ideas of Bela Julesz[10] - In the year 1962, Bela Julesz and J. Miller were able to show that a sense of depth can arise just from stereopsis. So they used several computer-generated pictures made up of randomly placed dots. Because the dot pictures did not contain any other information, like color or shapes, they could be sure that when their subject saw the picture it was a true 3D-impression (although the displayable „objects" were very limited). Note that they used a Stereoscope to view the pictures; therefore they found the basis for Autostereograms but did not invent them.

Putting two pictures into one [10] - Tyler and Clarke realized that it is possible to put the images for both eyes into one picture, regardless of the information. This is

the second step: Bela Julesz was just able to produce "random" images, usable for viewing through a Stereoscope, but now "real" scenes can be presented without technical support. The idea is to overlay both images with placing the dots carefully in a way that each serves simultaneously for two parts of the image.

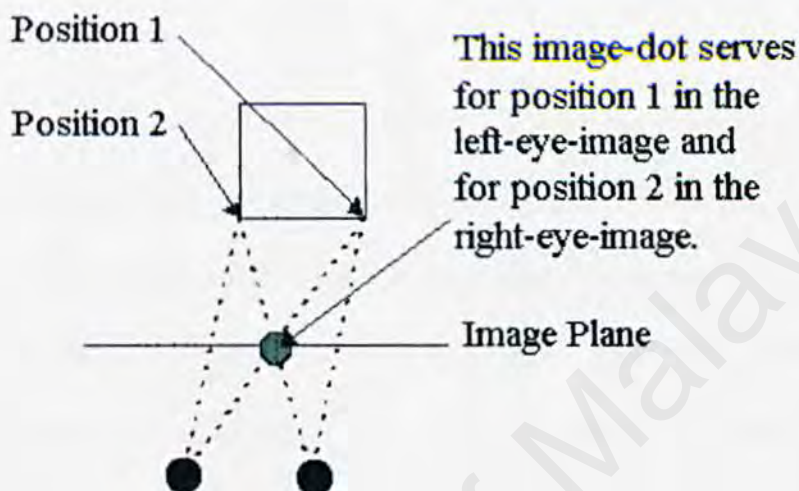


fig 1.0 : Overlaying the images for the left and the right eye into an "Autostereogram"[10]

1.3 What makes a picture 3-D?

A picture that has or appears to have height, width and depth is **three-dimensional** (or **3-D**). A picture that has height and width but no depth is **two-dimensional** (or **2-D**). Some pictures are 2-D on purpose. That's why they use only the most basic shapes. That's one of the basic differences between how 2-D and 3-D graphics are used: 2-D graphics are good at communicating something simple, very quickly. 3-D graphics tell a more complicated story, but have to carry much more information to do it.



Figure 1.1: triangles [8]

Take a look at the triangles above (figure1.1). Each of the triangles on the left has three lines and three angles. Its all needed to tell the picture is a triangle. We see the image on the right as a pyramid; a 3-D structure with four triangular sides. Note that it takes five lines and six angles to tell the story of a pyramid; nearly twice the information required to tell the block is a triangle. If making a 2-D picture into a 3-D image requires adding a lot of information, then the step from a 3-D still picture to images that move realistically requires far more [8].

1.4 What are 3-D Graphics?

For many of us, games on a computer or advanced game system are the most common ways we see 3-D graphics. These games, or movies made with computer-generated images, have to go through three major steps to create and present a realistic 3-D scene:

1. Creating a virtual 3-D world.
2. Determining what part of the world will be shown on the screen.

3. Determining how every pixel on the screen will look so that the whole image appears as realistic as possible.

1.5 Creating a Virtual 3-D World

A virtual 3-D world isn't the same thing as one picture of that world. Take a very small part of the real world -- your hand and a desktop under it. Your hand has qualities that determine how it can move and how it can look. The finger joints bend toward the palm, not away from it. If you slap your hand on the desktop, the desktop doesn't splash -- it's always solid and it's always hard. Your hand can't go through the desktop. You can't prove that these things are true by looking at any single picture. But no matter how many pictures you take, you will always see that the finger joints bend only toward the palm, and the desktop is always solid, not liquid, and hard, not soft. That's because in the real world, this is the way hands are and the way they will always behave. The objects in a virtual 3-D world, though, don't exist in nature, like your hand. They are totally synthetic. The only properties they have are given to them by software. We must use special tools and define a virtual 3-D world with great care so that everything in it always behaves in a certain way.

1.6 What Part of the Virtual World Shows on the Screen?

The screen shows only a tiny part of the virtual 3-D world created for a computer game. What is shown on the screen is determined by a combination of the way the world is defined, where you choose to go and which way you choose to look. No

matter where you go -- forward or backward, up or down, left or right -- the virtual 3-D world around you determines what you will see from that position looking in that direction. And what you see has to make sense from one scene to the next. If you're looking at an object from the same distance, regardless of direction, it should look the same height.

1.7 How to make it Look Real?

No matter how large or rich the virtual 3-D world, a computer can depict that world only by putting pixels on the 2-D screen. A number of image parts go into making an object seems real. Among the most important of these are shapes, surface textures, lighting, perspective, depth of field and anti-aliasing.

1.7.1 Shapes:

When we look at a 3-D graphical image on our computer monitor, we see images made up of a variety of shapes, although most of them are made up of straight lines. We see squares; rectangles, parallelograms, circles and rhomboids, but most of all we see triangles. A wide variety of objects can be modeled from a collection of polygons, and it is obvious that the surface has a polygonal skin [3]. However, in order to build images that look as though they have the smooth curves often found in nature, some of the shapes must be very small, and a complex image; might require thousands of these shapes to be put together into a structure called a

wire frame [8]. At this stage the structure might be recognizable as the symbol of whatever it will eventually picture, but the next major step is important: The wire frame has to be given a surface.

1.7.2 Surfaces texture:

In a 3-D graphic image, we can only look at the surface to get all the information possible:

- **Color:** What color is it? Is it the same color all over?
- **Texture:** Does it appear to be smooth, or does it have lines, bumps, craters or some other irregularity on the surface?
- **Reflectance:** How much light does it reflect? Are reflections of other items in the surface sharp or fuzzy?

One way to make an image look "real" is to have a wide variety of these three features across the different parts of the image. For realistic color, it's important for the computer to be able to choose from millions of different colors for the pixels making up an image. Variety in texture comes both from mathematical models for surfaces ranging from frog skin to Jell-o gelatin to stored "texture maps" that are applied to surfaces [8].

1.7.3 Lighting:

Lighting plays a key role in two effects that give the appearance of weight and solidity to objects, shading and shadows. Light source include ambient, spot, point, parallel and area. Ambient light is background light level that has color and intensity, but no direction [3]. The first, shading, takes place when the light shining on an object is stronger on one side than on the other. This shading is what makes a ball look round, high cheekbones seem striking and the folds in a blanket appear deep and soft. These differences in light intensity work with shape to reinforce the illusion that an object has depth as well as height and width. The illusion of weight comes from the second effect-shadows. One technique, called ray tracing, plots the path that imaginary light rays take as they leave the bulb, bounce off of mirrors, walls and other reflecting surfaces, and finally land on items at different intensities from varying angles [8].

1.7.4 Perspective:

If you stand on the side of a long, straight road and look into the distance, it appears as if the two sides of the road come together in a point at the horizon. Also, if trees are standing next to the road, the trees farther away will look smaller than the trees close to you. As a matter of fact, the trees will look like they are converging on the point formed by the side of the road. When all of the objects in a scene look like they will eventually converge at a single point in the distance, that's perspective [8].

There are variations, but most 3-D graphics use the "single point perspective" just described.

1.7.5 Depth of Field:

Another optical effect successfully used to create 3-D is depth of field. If you look at the object close to you, the object farther away will appear to be out of focus [3]. The use this depth of field effect for two purposes. The first is to reinforce the illusion of depth in the scene you're watching. The second reason is to focus your attention on the items or object they feel are most important.

1.7.6 Anti-alias:

A technique that also relies on fooling the eye is anti-alias. It computes the area of the pixel covered by a polygon [3]. Digital graphics systems are very good at creating lines that go straight up and down the screen, or straight across. But when curves or diagonal lines show up (and they show up pretty often in the real world), the computer might produce lines that resemble stair steps instead of smooth flows [3]. The computer can add graduated shades of the color in the line to the pixels surrounding the line. These "grayed-out" pixels will fool your eye into thinking that the jagged stair steps are gone. This process of adding additional colored pixels to

fool the eye is called anti-aliasing [8], and it is one of the techniques that separate computer-generated 3-D graphics from those generated by hand.

1.8 System Overview

1.8.1 System Objective

The objective of the project is to produce a 3D graphic interactive for the real world. This package will provide 3D environments – touring around FSKTM assisted by computer. The approach shall be interactive.

1.8.2 System Scope

The scope of the topic is to provide FSKTM environments – the real world – throughout a computer. The system is meant for FSKTM lecturers, visitors, students especially for the subject WRET 2106 – Rendering and Animation.

1.8.3 System Requirements

The system requirements are listed below:

1.8.3.1 Minimum requirements:

- Pentium 2 - 200MHz,
- 32 MB RAM,
- CD-ROM,
- Sound card,
- 3D video card.

1.8.3.2 Recommended requirements:

- Pentium 2 - 500MHz,
- 64 MB RAM,
- CD-ROM,
- 3D video card (minimum 16 MB),
- 3D sound card,
- Windows 98 / ME / 2000 / XP
- DirectX 8.1.

1.8.4 Systems Deliverable

Systems deliverable of the project are:

- The systems software of installation – 1CD,
- System documentation

- User Guide
- A PC.

1.9 Introduction of project – Interactive 3D Tour Around FSKTM

This project is the first project of producing **3D environments of the Faculty**. It also believes as the first ever in Malaysia. This project is basically to introduce the basic way of producing 3-Dimensional as the 3D it self is something new in Malaysia. The methods that will be used are texture mapping, bump mapping, and environmental mapping. This project being expects to be use as FSKTM first 'informational' facilities.

CHAPTER 2

LITERATURE

REVIEW

Chapter 2: LITERATURE REVIEW

2.1 Purpose

Before developing this project, researches were carried out on some of the education web site that consists of handbook section. Studies in this project also involved the evolution of 3d's and computer graphics also included studying on current software development tools. This section will discuss briefly about the literatures and survey that have been carried out through some findings such as articles available from Internet, library, senior's thesis although that there is not much on thesis about 3D's environments. Analysis on thesis materials also has been done.

2.2 The Study On Current Software Development Tools:

The production of a 3D model to represent a real-life situation is a difficult thing to do, given the complexity of the world around us. It is necessary that a number of measures be taken to ensure that the solution produced is a satisfactory solution to the problem. It has already been established that to provide a good solution 3D map is needed, however there are a number of ways in which a 3D map can be produced. This chapter are concerned with investigating the best way in which this project is created looking at a number of different methods of construction.

2.2.1 *OpenGL*

The OpenGL application programmable interface (API) allows users to create three-dimensional models using a set of primitives, including lines, points and polygons. With these primitives the user can create fully interactive 2D and 3D scenes. The creation of a scene is at a fairly low level for example scenes are built with points and lines rather than with high-level primitives. An expansion library called GLUT (The OpenGL Utility Toolkit) is used in conjunction with OpenGL to provide these high level primitives along with the following functionality:

- Callbacks to allow event driven processing e.g key handling or mouse movements.
- The use of multiple windows. OpenGL alone only allows one window per program.
- Fonts that can be used in the generated scenes.
- An "idle" loop and timers that can be used for animation.

3D Scenes are written in C++ using the OpenGL libraries and once complete, can be compiled like any other C++ program. A model produced using OpenGL is not dependent on a browser to display its 3D scene, meaning that the OpenGL code is also responsible for the handling of key and mouse inputs that may be used to navigate or manipulate the scene. A scene of equal standard to the VRML example would take some time to produce in OpenGL due to the way that OpenGL is used as an extension to the C and C++ programming languages. For example many of the features that are taken care of by the browser in VRML need to be addressed when

creating 3D OpenGL scene. This will make the scenes to become more complex but allows them to be far more versatile.

2.2.1.1 Advantages of the OpenGL Method

OpenGL is powerful and complete. Virtually anything one wishes to do with a 3D scene could be created (with the time and expertise) because of the way that OpenGL extends C or C++. High quality fast running scenes can be built that will run on any platform that has a C++ compiler, OpenGL and GLUT installed.

2.2.1.2 Disadvantages of the OpenGL Method

The creation of even simple 3D worlds is laborious and requires a fairly in depth knowledge of C or C++. This added to the fact that scenes must be compiled for the platform they are to be run on, makes them less portable and user friendly than other methods. There are very few software tools that can output OpenGL code, limiting the authors of such scenes to write them by hand. Bearing in mind the scale of the project, using the OpenGL method may be a poor choice.

2.2.2 VRML

Virtual Reality Modeling Language (VRML) was created specifically to aid the generation of 3D environments on the World Wide Web. Adapted from the Open

Inventor format. VRML was designed as an industry standard for describing 3D worlds or scenes on the Internet. VRML worlds allow all standard 3D modeling features such as texture mapping, lighting, and animation and user interaction. Scenes are built at a relatively high level using either a number of 3D primitives (box, sphere) or by the generation of specific custom objects using coordinate sets. The scenes (unlike other 3D languages) are viewed using either a stand alone browser or a browser plug-in. VRML worlds can also support embedded Java or Java-Script to increase functionality. Because VRML is so widely used (partly because of its similarity to Open Inventor) there are large ranges of development packages that can output VRML code, making the generation of a detailed 3D scene a more straightforward task. A simple example illustrating the power of VRML can be constructed quickly and easily as the browser software handles all values not addressed by the author. A fully lit navigable 3D world can be created using only a few lines of code.

2.2.2.1 Advantages of the VRML method

Simple VRML worlds can be constructed accurately in a relatively short time. The worlds that are created can be viewed by anyone with a compatible browser or browser plug-in.

2.2.2.2 Disadvantages of the VRML method

It would not be sensible to model a detailed location like the FSKTM by hand. Large

VRML worlds are usually created using an authoring package or world building software that can sometimes be costly or complicated to learn. The code produced by world building software is often very complex and difficult to modify, as even simple worlds can become swamped with indentation and bracketing due to the structure of the language. This may prove to be a problem if the world building software does not include all of the functionality that is required in the screen.

2.2.3 Java 3D

Java 3D is a programming interface specifically designed for the creation of 3D graphics in applets or applications. If the 3D model were to be created as a Java applet it would be possible to allow the model to be embedded into a web page. This would allow anyone to access the virtual world. However the Java 3D API is a fully-fledged programming library for Java (much like OpenGL) capable of producing photo-realistic output, and is fairly low level. Java 3D scenes have to be written from scratch with the Java code actually containing everything needed to show an image including the creation of the canvas on which it is drawn using primitives. Again, like OpenGL keyboard and mouse handling would also have to be written to allow any kind of user interact with the screen.

2.2.3.1 Advantages of the Java 3D method

As Java is an interpreted language and can easily be embedded into a web page a solution produced using Java 3D would inherit the portability of any Java program.

Low-level libraries would allow the author of a Java 3D world to control exact interaction and behavioral settings in the scene.

2.2.3.2 Disadvantages of the Java 3D method

Java and Java applets are well known for being slow at what they do. An effective 3D world would need to be responsive and give the sense of a continuous motion to maintain realism. In this case Java 3D would not produce an adequate solution for a large-scale virtual environment.

2.2.4 Clickable Image maps

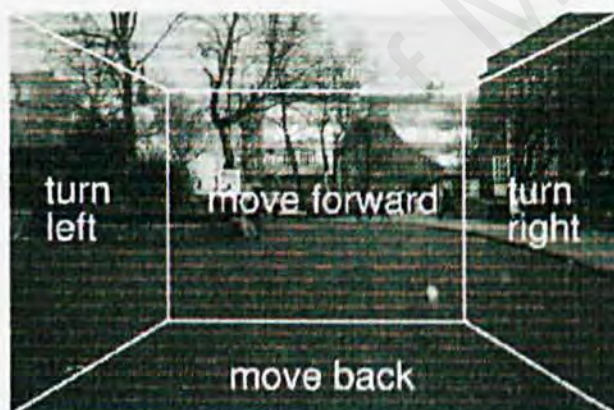


Figure 2.0: Clickable image maps

Clickable image maps are regularly used as an aspect of the World Wide Web (WWW) to allowing users to select different parts of an image to produce different outcomes. There are two types of image map which are used in accordance with

which best suits the application. Client-side image maps use the position of the users mouse, as calculated by the browser, to establish which area of the image was selected. Server-side image maps invoke the computation of where the user has clicked based on co-ordinates that are returned to applications. In both cases the position of the mouse, when clicked on an image corresponds to an area on the image map. The image map areas or ``hot-spots" each have a different action associated with them. These actions are usually used to redirect to a url or execute some JavaScript. The diagram for the main map view has an image map associated with it, with defined ``hot-spot" areas which link to corresponding expanded views. This method of defining co-ordinate ``hot-spot" areas could be used to make a pseudo-3D map by linking photographs together. The image map areas could be used to specify directions of travel as you walk through the scene. The image map is separated into four segments, each specifying a direction in which the user can travel from the current location. This is done in the HTML code by specifying the ``hot-spot" areas as shapes

2.2.4.1 Advantages of the image map method

By creating a map in this way it is easy to quickly create a first person walk through of the campus, using photographs. The map would be quick to build and easy to navigate, it would also represent the place accurately, and be an excellent visual reference. The map would also be platform independent, and easy to update and expand.

2.2.4.2 Disadvantages of the image map method

The small demonstration map only shows one point, and it takes 5 photographs to do this. The number of photos required to create a full map of a place would be immense. The photographs that are taken would have to be taken carefully to ensure that there is no overlapping of detail, and that they are all taken in similar conditions. Besides, photographs are directional, and to create a true 3D presence they would have to be taken from a number of different angles. For example, any route that is mapped will have to be photographed traveling both to and from a particular location.

2.2.5 3DGameStudio

The 3DGameStudio combines a high-end 3D engine, a 2D engine, a map and model editor, a script compiler (using C and C++) and huge libraries of 3D objects. It is user friendly as the GUI (graphical user interface) are easy to learn and quickly could make detail environments. The software is basically to produce 3D-Games but we could make use of the benefits of this software by producing this project.

2.2.5.1 Advantages of 3DGameStudio

The interface is user friendly even a beginner in design could make one. We could add our own C or C++ programming to manipulate the design, as we want the design to be interactive. This software comes with the Model Editor and Level Editor (will be discuss in later chapter). The level of detail could be high in this

software and could make this project real-time and we could feel the environment. This studio's could export other 3D's design from other software such as Maya, 3DstudioMax and other 3D's software.

2.2.5.2 Disadvantages of 3DGameStudio

This software however will using PC's RAM (Read Access Memory) to rendered all the design and it will required lots of RAM as the 3D's design will used lots of memory. If there's not enough RAM's, this software will stop render all the design and had to modified the design again all over to reduce the amount of design.

2.3 The Study on Other Existing System:

2.3.1 Virtual Los Angeles [<http://www.aud.ucla.edu/bill/UST.html>]

This project is the best design as it takes lot of artist to design the world. The UCLA Urban simulation team as a project to represent the Los Angeles Basin as a 3D virtual world created the Virtual Los Angeles project. The project is an excellent example of a 3D Virtual world. They created the scene from overhead photographs and from photographs taken on the street with a methodology that they themselves created for building a virtual world of this type. The building models were created from database information in sections and joined together to create a completed model.

“ Rather than attempt to build one large model from scratch, the Team has defined a methodology which allows multiple small models to be created and linked together. To date more than a dozen separate area models have been built, ranging in size from one to fifteen square miles. ”



Figure 2.1: Virtual Los Angeles

This project required lot of stuff as we can see detail of the texture. This projects however, only introducing fly-through environments as we could add some interactive elements such as car, door and anything else.

2.3.2 The 3D Internet Site Netrovia [http://www.netrovia.com]

This is another good example designing in VRML. This site introduce fly-through with click-able link, directed the link to other location of Internet site. Quite good on the interface with a 2D map (the town), so the user will not get lost when they refer to the map. They are using primitive block as their design. Lack of interactive as the design look dull for the system that needs the interactive stuff.



Figure 2.2: Virtual Shopping

2.3.3 WWW.STONYPLAIN.NET

This is an example of JAVA 3D. It combines the picture that had been taken in 360 degree. Well in visualization but no interactive and only shows picture of 360 degree in a single point. This design only good to use in quick review of a place.



Figure 2.3: Youth Bike Park

CHAPTER 3

METHODOLOGY

Chapter 3: **METHODOLOGY**

3.1 Building 3D worlds

There is no easy way for creating a 3D world, but there are some measures that can be taken to ensure that the virtual world that has been created will perform acceptability in terms performance and speed. It's important to follow four simple steps to make sure the worlds they create are accurate and well built [11]:

- Use orthographic views for modeling and perspective views for rendering to ensure that the dimensions and detail of the scene is maintained
- Use primitives and instantiate shapes as much as possible for memory efficiency
- Leave surface details to surface maps and simplify model geometry, detail that will not be seen wastes valuable system resources
- Leave out the details you can do without, don't use unnecessary rendering effects or animations.

3.2 Recognition of Problem

FSKTM is situated in a large city campus, with environmental environments around the building in University Malaya. It is more complicated to draw 3D environments due to it will cost lots of memory for a computer to rendered to scene. Finding a particular building or rooms may be a simple task for someone with

knowledge of the faculty, but to someone from outside of the faculty, this may be more difficult.

The current means provided to help visitors find their way around the faculty are not sufficient. It is clear that a more up to date method of locating buildings within the city campus is needed.

3.3 Telepresence

The notion of telepresence when related to the field of 3D visualizations is used to describe how much a user feels physically present in a remote world or scene [11]. For a user to become involved and immersed in a virtual world it is essential that there is some level of familiarity about the scene that represents what could be a real place. The use of representation as a physical location makes the notion of telepresence easier to create as the user is automatically forced to visualize the location that is being represented [11].

The introduction of photos of locations around the campus can help reinforce the feeling of telepresence for the user [11]. A good notion of telepresence in a world can be gained even if the virtual world is not 100% accurate to the real world. Items in the world can be created without using specific measurements as long as they are in general proportion to other objects in the scene.

To create a notion of telepresence in the FSKTM interactive faculty map is make sure that all objects in the world looked consistent with their respective physical locations, their photographs and other objects in the scene. This method of building the model by looks alone may not provide the most accurate 3D map, but it will provide a good visual reference that the user finds easy to relate to.

3.4 Primary Sources

Photographs were taken around the faculty (FSKTM) as main source of information for creating the 3D scene. Over 100 photographs were taken to model the detail of the faculty. These were taken on site in overcast weather conditions. Some photographs were of complete buildings or departments, while others were merely detail shots allowing seeing how particular buildings are joined together or look from particular angles.

The other source that have been using is thorough Internet. I had spend almost all the time on Internet due to lack of book about how to build 3D environments in faculty, plus the lecturer it self. The research on existing system had been done thorough Internet, and all the skill needed to design in this project had been learned using Internet.

3.5 Secondary Sources

As I did not have access to certain room of the faculty, I relied mainly on photographs and sketches to create the 3D scene. The uses of faculty maps however make a number of diagrammatical campus maps to assist with placement and rotation of buildings in relation to others.

3.6 Project Scale

Due to the size of the faculty site it is important that measures are taken to simplify the 3D scene. A "happy medium" between level of detail and accuracy will have to be reached.

3.7 Questionnaires

This method will be used to find out important role of 3D-environment to the students it self. Listed below are main question for the questionnaires:

- Did they realize important of 3D?
- Is there any 3D's environments had been introduces to them? Where?
- Did they have been thought how to design 3D?
- Did they agree to have 3D-Virtual faculty, as faculty's informational?
- What is the level of their 3D's knowledge?

3.8 Project Development Life Cycle

In order to develop a system in a organized and effective way, it is necessary to follow a sequence of steps to accommodate a complete set of tasks, which is generally called a process. A process is referred to a series of steps involving activities, constraints and resources that help in produce a good system. Thus, it usually includes a set of tools and techniques.

When the process involves the building of certain kind of product, it is referred as a life cycle. Therefore, the development process of this system is defined as the Project Development Life Cycle, because it describes the life of this system (which is the output product) from its conception to its implementation, use and maintenance or upgrading.

3.8.1 Software Process Model

The software process model used to develop this project is based on the waterfall model with prototyping as shown in figure 3.0. This module is best suited to develop large and complex system. It is used because this system will be develop under the environment that needs to be separated into different process phases, which cascade from one phase to another. This model consists of several phases; requirement analysis, system design, coding, unit and integrations testing, system testing, acceptance testing, operation and maintenance.

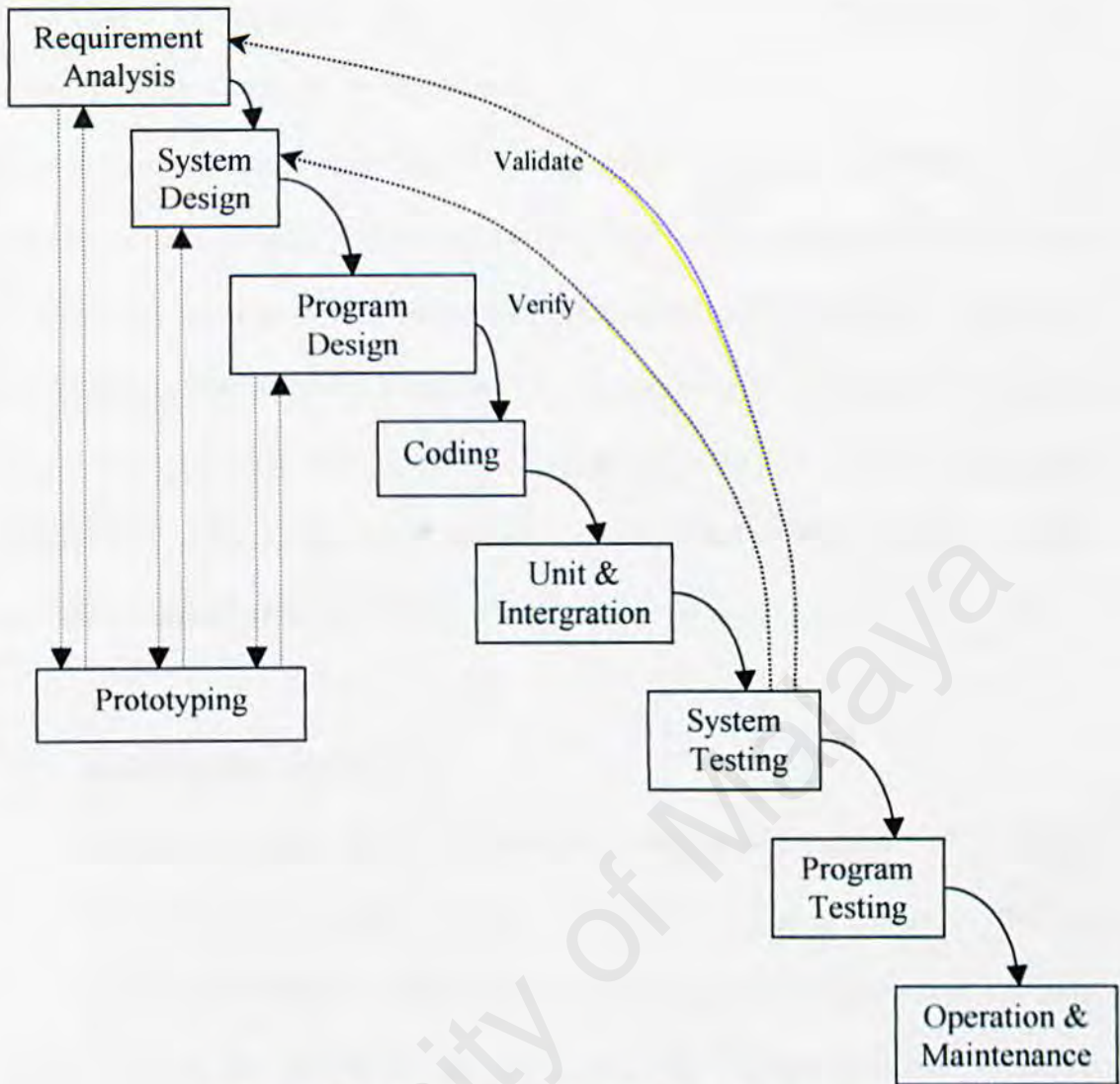


Figure 3.0: The Waterfall Model With Prototyping

The prototyping is incorporated into the waterfall model because this system is a system that makes it vital to test out the functionality of its module of the development process gets into the implementation stage. The stage and prototype will also allow potential user to test out this system and any necessary modification can be made before it is implemented. Another reason why the waterfall with

prototyping approaches was used is it offered a menu of making the development process visible compare to another model.

Throughout this model, this system interface is built and tested as a prototype, so the user understands what the system will be like. This, major kinks in the requirements are addressed and traced well before the requirement are officially validate during system testing. Prototyping is useful for validation and verification. Validation ensures that the system has implemented all the requirements, so that each system function can be traced back to a particular requirement in the specification while verification ensures that each function works correctly.

- **Requirement Analysis**

In this first phase, all the information regarding this system such as survey process will be gathered through internet, reading material and by interviewing lecturers and students. The requirement of the system has been listed out by discussing with my supervisor. The software and hardware requirements for the project have been determined and the research on the existing system has been carried out too.

- **System Design**

The system design involves drafting out data flow diagram shows how system function react one to each other. User Interface is designed to show how exactly the system will be look like. Prototyping is used in the phase

together with the waterfall model to reduce the uncertainty about what the system should do.

- **Program Design**

This is the stage where the computer programs are created. The design must be translated into a machine-readable design.

- **Coding**

All interactive models or objects will be coded using selected template in 3DGameStudio or design own coding to make the model or object react.

- **Unit and Integration Testing**

This system is realized as a set of program units at this stage. Unit testing involves verifying that each unit meets its specification. If the unit testing fail, the system prototype has to be redefined or this system and program design has to be reprocessed. Once the unit testing is carried out successfully, the individual program units or programs of this system, which had been developed, are integrated to make sure that they work properly when joined with other.

- **System Testing**

The system testing involves a test of the whole system to ensure that the function and interactions specified initially have been implemented properly. This system is compared with the specified requirements. Validation and verification have been taken in order to check that the system serves its intended purpose.

- **Acceptance Testing**

Besides the test that had been run by the developers, the system also test by a use, making sure that its meets their understanding of the requirements, which may be different from the developers.

- **Operation and Maintenance**

The system is installed and put into practical use. Maintenance is carried out to correct any errors that were not discovered in the earlier stages, enhancing the system functionality and also improving the system units.

3.9 Justification of Proposed Methodology

Below are the convincing points and strategies in applying the propose methodology into the system development:

- Theoretically, one development stage should complete before the next begins. In overall, the propose methodology presents a very high-level view of what goes on during development and it suggests me the sequence of

events that I expect to encounter. However, in practice, these stages overlap between one and another, and feed into each other. For instance, during program design phase, problems related to system design would be encountered. Therefore, it is more suitable to say that an application system development process is not a simpler linear model but involves a sequence of repetition of the activities.

- Associated with each process activity will be milestone and deliverables, so that I can use the model to estimate how does the project is to completion at a given point in time.
- In addition, this model is expected to help me lay out what the system needs to do in an order way. Its straightforwardness and simplicity make it easy to explain to others who are maybe not familiar with the system development.
- Prototyping conducted currently with the requirements analysis and definition stage of the development life cycle, effectively helps in ensuring the developing system time to time that is always meets its definition of needs and is feasible enough. If any error or inappropriate output and definition found during the early stage, correction and improvement will take place immediately. It is therefore able to avoid or at least reduce the cost of changing the whole testing stage when everything almost comes to final stages.
- Prototyping is useful for verification and validation makes sure that the right system being built, whereas verification checks the quality of the implementation.

CHAPTER 4

SYSTEM ANALYSIS

Chapter 4: SYSTEM ANALYSIS

4.1 Requirements Analysis

System requirement define the needs and services by the system. It provides a guideline for developing in the system in a complete manner. There are two categories of requirements are functional and non-functional.

4.1.1 Functional requirement

A functional requirement is any function of feature that must be included in the system to fulfill the users needs and be acceptable to the users [12]. These functions describe the interactions between the system and the user and also the action of the system when certain stimulus is provided. The functional requirements for this system are as follows:

- **User Section**

- User Termination Menu

This menu allows the users to quit the program when they want to stop using the system.

- Get Help Menu

This menu allows users to get help whenever they face with any problems.

4.1.2 Non-Functional Requirement

Non-functional requirement are features, characteristics or attributes of the system as well as constraints that limit the boundaries of the system [12]. Any constraints usually limit the system in term of the selection of programming language, platform deployment and implementation.

4.1.2.1 Reliability

The degree in which the system operates in a user acceptable manner when used in the environment for which it was design and does not generate dangerous or costly failure when it was design and does not generate dangerous or costly failure when it is applied in a reasonable manner.

4.1.2.2 Efficiency

The system should be implemented in the most efficient manner with the least requirement for the system to run smoothly.

4.1.2.3 User Friendliness

The system should be provides all the follow of user friendliness:

- Control

The system should provide user manual, as the new user will know how to control and interact with the system. This system should be standardizing.

- Info

This system will provides user with info of object or anything in the interface. The info should be interactive.

- Expandability

This system will in a CD installation and the cost of the CD's is acceptable for the user.

4.2 Hardware Requirement

Computer hardware is one of the most important things to take care with.

The least need of computer software for the software runs smoothly are listed below:

- Pentium 3 – 500MHz
- 64 MB RAM – for render about 5000 portals.
- 3D Video Card (+16 MB)
- CD-ROM
- Sound Card

4.3 Software Requirement

This section will discuss of software that will be use to produce this project.

4.3.1 Operating System (OS) Software:

The OS that will be use is Windows XP. This OS is the latest product from Microsoft Company and it supports all type of software. Plus, the stability is so good

and hard to “crush”. This will benefit when render any 3D map or object because it takes lot of time to render any of 3D design.

4.3.2 Software Development Tools: 3DGameStudio

The name, 3DGameStudio tells that producing 3D game, but could implement the benefit of this software of producing this project. The project: “Interactive 3D Tour Around FSKTM”, tells that need have interacted between user and the system itself. This software is object-oriented designing, plus, could program in C or C++ the interactivity such as opening door and more by the programmer itself. This software also support big range of 3D’s design and could make realisms onto the project.

“An excellent toolkit to quickly prototype and develop 3D graphics applications”[13]

Dr.Dobb's Journal.

4.3.2.1 The Editors

This software comes with two editors, Level Editor and Model Editor. The Level Editor is for designing world interface and Model Editor for designing model interface. The level, model and terrain editors make creating landscapes, placing light effects, defining movement paths and modeling actors a straightforward process. Each level consists of an environment with fixed and moving parts - landscapes, dungeons, buildings or cities. Within the environment you can place **colored light sources**, which throw real shadows, and 3D sound sources, which

emit background sounds. Textures and Sprites can be created with any paint program. Animated 3D models are created with the model editor-MED.

This software could import levels and animated models created with popular 3D game editors like Worldcraft™ or Milkshape™. Usual 3D file formats (**X, 3DS, MDL, MD2, MAP, WAD**) are supported. Terrain can be imported from freeware terrain generators like **Terragen™**. If you are used to high-end editors like **3D Studio MAX™**, you can create your animated models with the GameStudio/MAX plugin, which is available for MAX 2, 3, and 4. Through the Milkshape™ compatible filter interface, the model format of almost any game can be imported or exported.

4.3.2.2 The Engines

The game engine is the core of the development system - it generates the 3D effects and controls the artificial intelligence of the actors. Due to its combined BSP-Tree and terrain rendered, the A5 engine handles indoor and outdoor sceneries equally well. It has a lighting engine that supports true shadows and moving light sources. The sophisticated threefold culling algorithm renders even huge worlds of 100,000s of polygons with over 70 fps. Programmers can use the DLL interface for adding new effects and features.

4.3.2.3 Model Editor (MED – model editor design)

The Model Editor is known as MED. MED is used to create or convert the 3D models and terrain for the A4 and A5 engines of 3D GameStudio. It has an easy to use interface that is ideal for basic 3D model creation and skinning. This MED comes with manual so that it isn't hard to learn and manipulate. It comes with wide-range of term like listed below:

- **.MDL, .MD2:** Soft skin model file formats, used by major commercial game engines. Both model formats can be imported. MED converts them into the A5 engine's native **MDL4** format, which also uses the .MDL extension. Stored in these files are the triangles, skin mapping information, the skin image, the animation frames, bones, generally everything to do with the model.
- **.X, .3DS, .ASC, .WMP:** 3D file formats, exported by most 3D modelers. X is the Microsoft model format; WMP is the GameStudio level format. ASC and 3DS are triangle lists that describe the shape of the model, without skin texture or mapping. All 4 formats can be imported; .3DS and .ASC can also be exported.
- **.PCX, .BMP, .TGA:** These are the palletized or true color images that can be imported as skins for models, as well as height maps for terrain. TGA files contain an **alpha channel**, which gives a transparency value for each single pixel of the skin (A5 extra edition or above). MDL files contain one or

several skins. In MD2 files skins are not included, but referenced by file name (see also skin).

- **.HMP** (heightmap): Gamestudio's terrain format. A height map consists of a rectangular mesh of triangles. Height maps can be imported from PCX or BMP height images that are produced by terrain generating software. All better terrain generators support heightmap images. A height image can be painted with each paint program as a grayscale picture: The brighter a pixel, the higher the corresponding location. Heightmaps can also be converted to models.
- **Animation:** A Sequence of Frames. See Frames.
- **Face:** see Triangle
- **Frame:** A set of vertices that define a single pose for a model. Frames have names, which can be used in the script language to select a single frame. A number of connected frames are naming a **Scene**.
- **Heightmap:** see HMP
- **Interpolation:** A method that calculates the position of the vertices in a model between two arbitrary frames. The A5 engines of 3D GameStudio support mesh interpolation.
- **Mesh:** The collection of triangles that defines the shape of the model itself. It essentially is the model. A5 can handle arbitrary complex models, but for performance reasons the mesh of an average model in a 3D game should not consist of more than 300 to 800 triangles.
- **Model:** see Mesh

- **Normal:** An imaginary line that points out perpendicular to the triangle, i.e. in the direction that a triangle is facing. A triangle is only visible from its normal side, and invisible from its backside.
- **Polygon:** See Triangle.
- **Scene:** A set of frames that make up a sequence. For example, "Attack" or "Death". Normally the frame names of a scene only differ in a number.
- **Sequence:** See Scene.
- **Skin:** The picture, or texture, that is wrapped around a Model. Skins can be imported in .PCX, .BMP, or .TGA format, 8 bit, 24 bit or 32 bit with alpha channel. Unlike other engines, A5 is not restricted to 256x256 skin sizes. The skin can be of any size, and can be arbitrarily mapped onto the model parts.
- **Skin Linking:** When a model uses an external .PCX file for one of its skins (MD2 format only), the .PCX file is linked to the model.
- **Skin Mapping:** The 2-dimensional triangle mesh that defines which parts of the skin are wrapped to which parts of the model.
- **Terrain:** see HMP
- **Texture:** See Skin
- **Triangle:** A flat triangle that is defined by three vertices. A **Mesh** is made up of these. Some old 3D cards (Voodoo) can only display triangle sizes of 256x256 pixels maximum.
- **Vertex:** A point in 3D space, represented by three coordinates, its X, Y and Z position.

4.3.2.4 World Editor Design (WED):

WED is a map editor for the 3D real-time game engine A5, and part of the 3DGameStudio package. It is based on the popular Qoole level editor, written by Matthew Ayres and Paul Hsu. WED is an acronym for 'World Editor'. It is capable of creating new A5 levels and importing existing levels saved in Qooles ".qle", id's Quake ".map" or GameStudio's ".wmp" file formats. WED features the object oriented level editing methodology, as well as an intuitive user interface that provides powerful editing functions. These functions include:

- Object oriented grouping and scooping
- Object move, rotate and scale manipulations
- Prefabricated block objects
- Block vertex, edge and face manipulations
- Block hollowing and subtraction
- Wire frame / Solid Polygons / Texture previews
- Easy map navigating interface
- Texture move, rotate and scale manipulations

WED allows you to create 3D game levels effortlessly without requiring you to have any prior 3D editing experience. Both beginners and advanced level designers will find WED to be an indispensable tool for creating professional looking 3D game levels quickly and easily.

4.3.2.4.1 Map Editing Concepts

There are 2 primary elements in a 3D level map: **blocks and entities**. The former is simply a 3 dimensional solid block. In a 3D level, **the world is made of solid blocks**, or objects consisting of blocks. To make a simple **cubic room**, one would need 6 blocks to enclose off an empty area; 1 block for **the ceiling**, 1 block for the floor, and 4 blocks for the surrounding walls. As for the second element to a level, there are 3 kinds of entities: map, model, and sprite entities.

4.3.2.4.2 Object Oriented Concepts

WED considers everything as an object. An object can be an entity, a convex solid block, or a group of other objects. This simple concept results in the powerful feature of multi-layered grouping. Grouping allows you to build structures such as a staircase, an arch, a bridge or a chair out of simple blocks. The Object Oriented editing methodology hides unnecessary information from the user. With it, you can construct a level based on logical objects instead of thousands of blocks and vertices. The methodology also introduces modularity and portability. The user can build independent sections of a level without worrying about the rest of the map.

A 3D game level - a **Map** - is built from 3D objects that consist of basic elements named **blocks**. Image patterns, named **textures**, are mapped onto the surfaces of the blocks. Blocks can be of any shape, and their material can have certain properties, like being passable or impassable. The whole system is named "Constructive Solid Geometry" (**CSG**).

How impressive the level will look depends on the lighting. Lights and shadows are an important feature of the **A5 engine**. Right combination with colored and white lights, they can really add atmosphere to a level.

The WED editor saves and loads maps and prefabricated 3D objects as **WMP** files. The textures are done with each paint program and then stored in **WAD** collections by WED's texture manager. Within a map, further maps can be placed, as well as other objects like **models** or **sprites**. The **Engine** - this is the 'core' program, which runs the game and displays the 3-D world on the screen - needs the level in a final **WMB** format that is compiled from the WMP file and one or more WAD texture resources if you click onto the **BUILD** button.

In order to interact with the system, like doors or monsters, you need a **WDL** script to describe its behavior (if you not prefer to program anything in VC++). WDL scripts are also responsible for releasing the **special effects** built into the A5 engine, like dynamic lights, flares, particles, fog, and polygonal sky. Many scripts are prefabricated and included.

The following diagram is the workflow and the elements from which any 3D design can be created with 3DGameStudio.

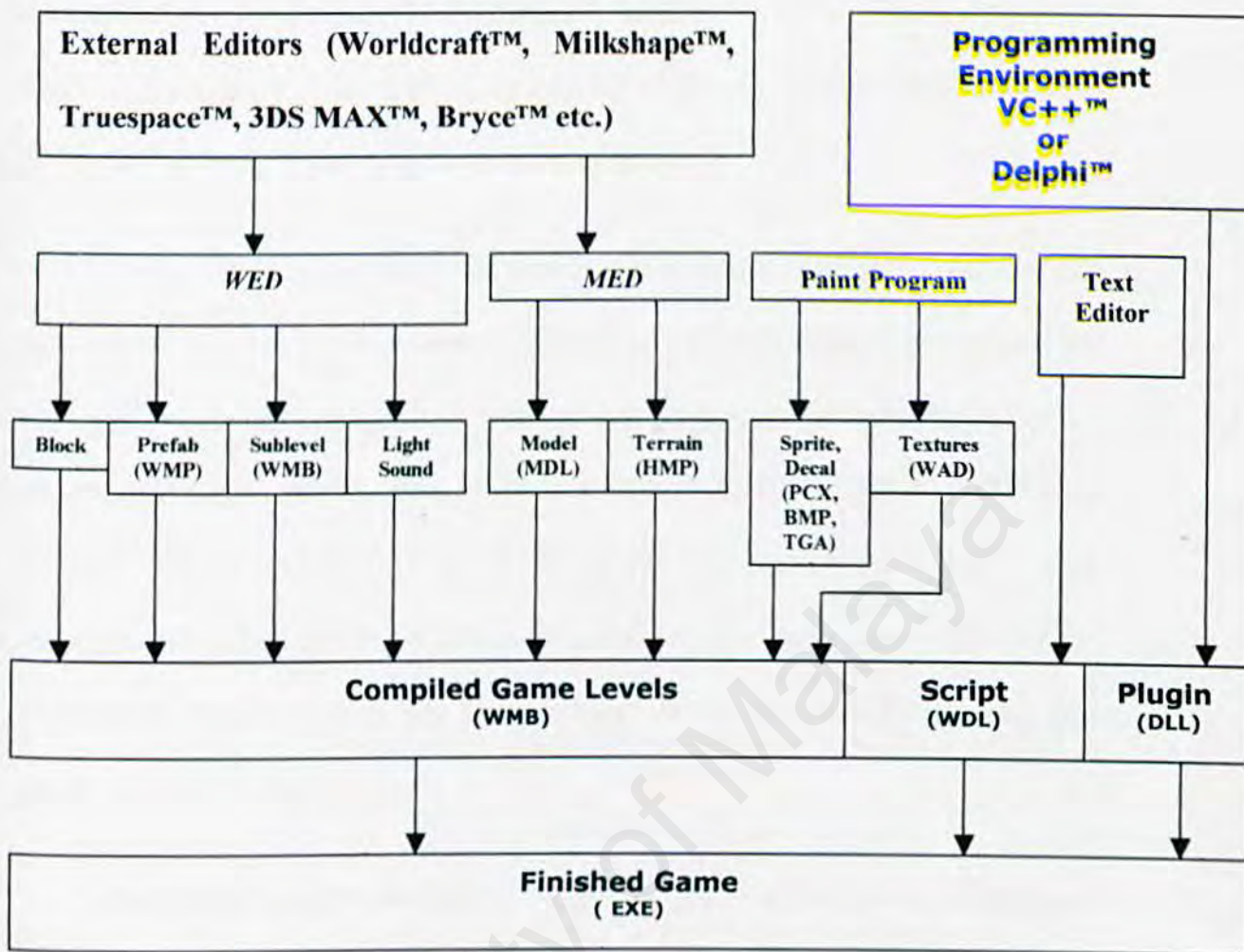


Figure 4.0 : Work Flow of 3DGameStudio

4.3.2.5 Rendering 3D Design

Compiling a map can take WED from some seconds up to many hours or even days. It depends on the design of the map, and on the number of blocks and lights used. It's a square dependency - that means if you double the number of blocks, building normally takes four times as long. An indicator for the build time is the number of **portals** displayed during the build process. Portals are the planes that

separate the **regions** (sectors the BSP process splits the level into). Each surface of a block can generate a splitting plane, and thus produce one or more portals. Sealing a block can increase, or decrease the number of portals.

If the design more than 5,000 portals, try to reduce your level – for example by replacing prefabs by map entities. Otherwise the build process would take too long, and would need hundreds of MB's of hard disk space. If we build extremely huge levels, the PC should be equipped with a lot of RAM - at least 128 MB for up to 10,000 portals, and 256 MB for 15,000 portals. The preview mode builds much faster, but the WMB file is not optimized, resulting in a lower frame rate, and the shadows are not so smooth. The shortest build time is achieved by marking **Test Map** and **Preview** simultaneously.

Sometimes when renderings or compiling, you'll be encounter warnings, or even error messages when building the map. In the latter case the WMB file won't be produced. The errors are usually caused by problems in your maps. Mostly, the map designs had exceeding the map limits.

4.4 Techniques Used to defined Requirements

- Library Research

During the development of this system, I spent lots of time in the library of FSKTM to look for the computer books regarding with design systems, design method and existing system.

- Observation

I come across many systems that evaluate the same system that I want to build. From those observations, I get better ideas about how to build the design interactively.

- Discuss with Peers and Lecturer

I get several useful advices from my supervisor during each week section meeting and presentation. It is useful for error corrections and act as reminder when I carried out the system development process.

- Thesis Documentation Analysis

I did study some past year thesis documentation in order to know how to produce the thesis report. From the documentations, I also gained better knowledge about the software development process.

CHAPTER 5

SYSTEM DESIGN

Chapter 5: SYSTEM DESIGN

In this chapter, the system design of this project will be discuss. The overview of the system is presented followed by a description of the separates modules in the system.

5.0 Overview

System design or physical is considered as an important part of the software development process where the requirements of the system are translated into the system characteristics to meet the user requirement and satisfaction. It is utilized an applied regardless of what kind of development model or standard being used. The system design includes a complete description of the functions and interactions involves. They are a series of activities involved in system design process, which are analyzing, coding, designing and testing the system or prototype to ensure that it confirms to the software specifications and requirements, which have been define in early stages.

In this system design phase, I will cover the following issues:

- System Architecture Design
- System Functionality Design
- User Interface Design

5.1 System Architecture Design

The system architecture for this system is chosen based on the scope and complexity of this project. In general, the system functions in a two-tier architecture. This architecture is chosen because it allows any part of the system to be modified without change the other part of the system. Thus, it allows the system to operate at its highest efficiency.

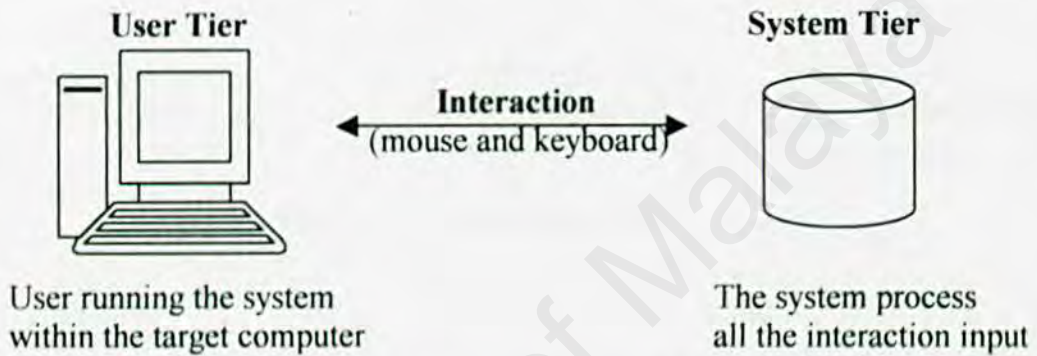


Figure 5.0 : System architecture of the system.

5.2 System Functionality Design

During the system functionality design, I seek to specify the structure and operation of program that will meet the requirements articulated during the information processing system design phase of system development. System functionality design transforms all the requirements into an organized picture of the system functionality and data flow diagram.

5.2.1 System Structure Charts

The objective of system structure chart is to show how the modules in the system are related to each other.

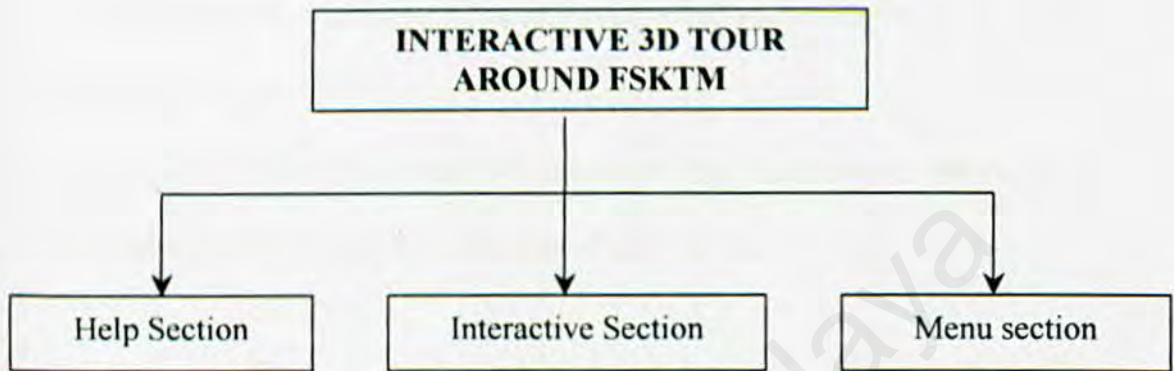


Figure 5.1 : Structure Chart

In general, the system consists of three major parts, which are the Help Section, Interactive Section and the Menu Section. Basically, the Help Section is to provide manual of key interact during the system being run. The Menu Section lets the user choose what they want to do; either quit or seeing the manual of the system. Finally the Interaction Section is the key of the system where user interact with the system interface; either moving around or does something like open a door.

5.2.2 Data Flow Diagram (DFD)

In the process design, the system is designed based on the data flow diagram. Data Flow diagram (DFD) is a graphical technique used to depict the flow of user's key input for the system to process and react. It gives an overview of the system's inputs, processes and outputs.

The DFD specifies the system at the logical level (what the system does) rather than the physical level (how it does). The use of DFD has several advantages:

- It is easy to understand since only four symbols are used.
- It provides a better understanding about the relationships between the system and the sub-system.
- It helps to identify the required command and processes of the proposed system and making sure that they have been defined.

DFD is easy to understand since only four symbols are used: processes, data flows, data stores and external entities. These symbols help to specify the physical aspects of implementation.



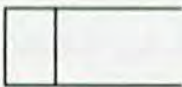

Symbol Name	Process	Data Flow	Data source	Entity
Symbol				

Table 5.2 : Gane and Sarson Notation of Data Flow Diagram

C.Gane and T.Sarson base the convention, which is used to design DFD on the work. The data flow is conceptualized with a top-down perspective. So the Context Level Diagram will be drawn.

5.3 User Interface Design

The user interface is a vital component of the system. For most users, the interface is the system. There is several kind of user interface. They are natural-language interfaces, question-and-answer interfaces, menus, form-fill interfaces, command language interfaces and graphical user interfaces (GUIs). The user interface has two main components: presentation language, which is the computer-to-human part of the transaction, and action language, which characterizes the human-to-computer portion. For the system, I had chosen graphical user interfaces (GUIs) as my interface.

5.4 DESIGN PROCESS

The design process is one that takes verbal and written requirements and specification of the program and transforms them into representations of the programs with prototype. In other word, the process translates abstract ideas into software modules. There are many strategies of techniques for performing systems design including model-driven, prototyping, RAD (research and development) and object-oriented design. These strategies are often combined to harness the best feature of each into a single strategy.

CHAPTER 6

SYSTEM TESTING

Chapter 6: System Testing

Introduction

To produce a quality system – with fewer error and which work according to specification and performance requirements, system need to be tested. The purpose of testing is to detect the errors that have not been discovered yet. All tests should be traceable to user requirements too. That's mean the software must meet all the requirement of the user. Test data are being input into the system for processing and the results examined. A number of users are given the opportunity to try the system so as to trace any uncover errors or misunderstandings before launching the system.

6.1 Testing Strategies

Testing Strategy is a strategy of establishing the existence of errors. It is also a general approach to the testing process rather than a method of devising particular system or component tests.

The testing strategies consists of following:

- Top-down testing

Testing starts with the most abstract component and works downwards until all the modules are tested.

- Bottom-up testing

It is one of the popular approaches used to test large systems. Testing starts with the fundamental components and works upwards.

- Back-to-back testing

Used when versions of a system are available. The systems are tested together and their outputs are compared.

- Thread testing

Used for system with multiple processes where the processing of transaction threads its way through these processes.

6.2 System Testing

System testing can be divided into unit testing, integration testing and system testing. Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. A good test case has a high probability of finding an undiscovered error from the system.

For this system, unit or module testing and top-down testing have been carried out for integration testing.

6.2.1 Unit Testing

The primary goal of unit / module testing is to confirm that the unit is correctly coded and that it carries function it is suppose to carry out. It is the initial testing stage for the completion of each component class. All the logical error that contain in the classes will be detected. The interaction testing between components are initially avoided and to carried out later in the bottom-up integration testing.

6.2.2 Integration Testing

Integration testing is to used to ensured that the system will work correctly when all the models and script files combined together. Each model is tested individually first to see the resultant of combining together with the scripts. Then it will be tested to whole system to completing the system. This approach will repeat until all the models being tested.

6.3 Conclusion

Testing is the most important steps in developing the system perfectly. Precision and accuracy of output data or appearance is considered during this process. Unit, integration and system testing has been carried out for this systems. The objective of a system will only achieve after all the through testing done by different user with different aspects.

CHAPTER 7

SYSTEM

EVALUATION

Chapter 7: SYSTEM EVALUATION

In this phase, this system was evaluated to identify its strengths, limitations, and proposals were made for the future enhancements.

7.1 Problems Encountered and Solutions

A number of problems were encountered throughout the development of this system.

7.1.1 Difficulties In Determining The Scope Of The System

It is impossible to build a full-scale complete faculty in this system within the short time frame. Not all the particular rooms or buildings are put in the system. Advices and opinions were given by project supervisor to outline the scope of the project to build during initial stages. On the other hand, the results of survey and interview have given an outlook of the system scope.

7.1.2 Difficulty In Designing Telepresence Environments

Without much experience in such designing a telepresence environments, standard and user-friendly interface, makes designing phase become more difficult and challenging. To gain more information on the designing phase, lots of interesting web sites and lots of “try and error” were taken as reference for developing a proper and standard 3d’s design.

7.1.3 Difficulty In Rendering The System

The software however needs lots of PC's RAM (Read Access Memory) to render the environments in the system. It takes about 64 RAMs to render only 5000 portals; that is a small number of portal compare with the system needed.

7.2 System Strengths

There are several advantages of this system as listed below:

7.2.1 User Friendliness

The interface of the system is user friendly and easy to navigate and understand. Besides, this system being included with manual; within hardcopy or through the system help menu

7.2.2 Easy To Use

This system is very easy to use. The commands and the layouts are simple and well organized. Therefore, it is easy to learn up, use, navigate and remember a certain room in the faculty.

7.2.3 Easy Accessible

This system is offline software and can be download or install in a PC. It can run on almost all PC in the world right now; either MAC or Windows.

7.2.4 Real-Time Rendering

This system can consider fast enough if we compare with other 3d's player, such like VRML or OpenGL. System interface have been in real-time rendering, where its become more realistic than others authoring tools. The movements of the system are so smooth; without have to render again the environments.

7.2.5 Free From 3rd Party Platform

Unlike other software developments tools, these systems are stand-alone player without have to download other's platform for running the system. Thus, its makes the user just 'click and run' the programs and it have been prove that so conveniences to the user.

7.3 System Limitations

However, there are limitations in this system that are not resolved yet.

7.3.1 Lack Of Detail Structure

The system could not being include lot of structure due to PC limitation on designing phase. Detail structure in a system mainly could not being build with ordinary PCs like Windows, due to lack of memory usability and performance of the PC's processor.

7.3.2 Limited Functionality

This system only provides through offline usage. User can only download and install the program through their PCs. It could not runs as web-base application due to the bandwidth of Internet connection and the associated files that needed the system to runs smoothly. Therefore, the system cans only runs by user's PCs.

7.4 Future Enhancement

The system should be maintained throughout the lifetime of the system because the user requirements might vary from times to times. Enhancement in the future will extend the usability of this system. Moreover, the system limitations should be improved to enhance functionality.

Here are some suggestion and possible future enhancements:

7.4.1 Enhance User Interface

User interface should enhance from time to time. Multimedia elements such as streaming video, more graphics especially animated graphics or animated entities should be added to increase its attractiveness, impressive and interactive and to make the system multimedia map program.

7.4.2 More Functionality Added

Dynamic key configuration should be added to make sure the system more users friendly and flexible for user to change the configuration.

7.4.3 Develop For Other Platform

The system should be design web-base to make sure that everyone in the world could use the system as information of the faculty.

7.4.4 Provide In Other Language

Since the usage of this system will be used world wide, other language should be an implement to the system for different user's language.

University of Malaya

CHAPTER 8

DISCUSSION

Chapter 8: DISCUSSION

In this chapter we will discuss what was happen in the development phase of the system my opinion about this project.

8.1 Modeling and Animation

I have to finish modeling and animations as soon as possible to make sure that I did not exceed the timeline in development phase. Due to time constrain, the animate character could not be detail. Below is what I have finished before the programming stage start.

8.1.2 Modeling

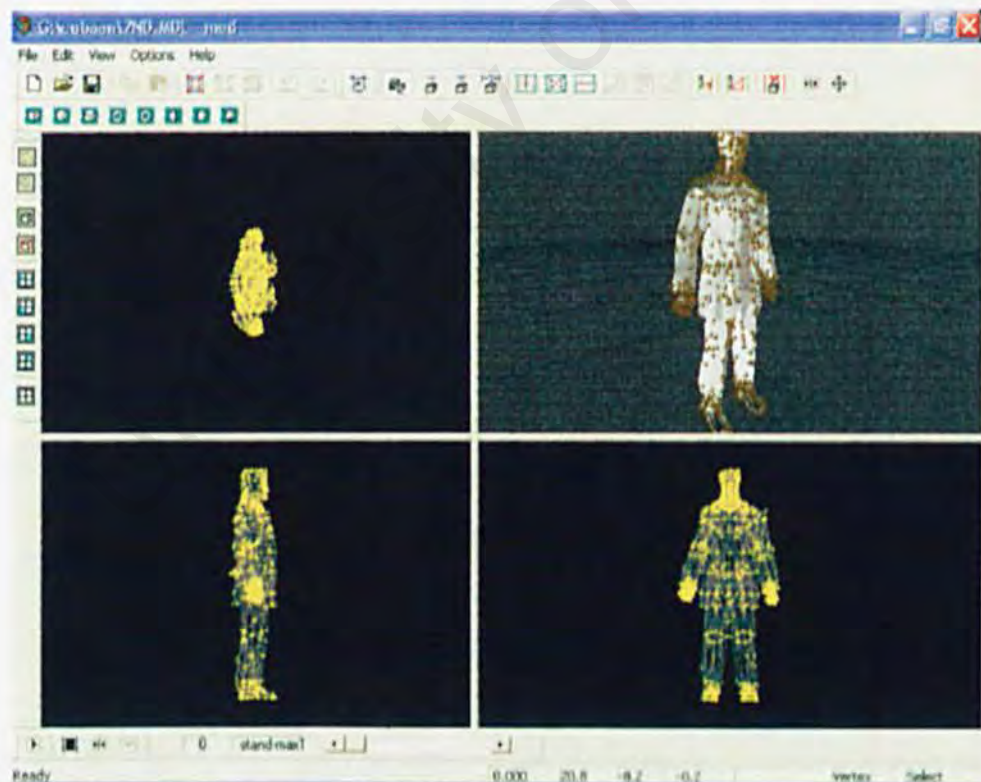


Figure 8.1: Model for represent user.

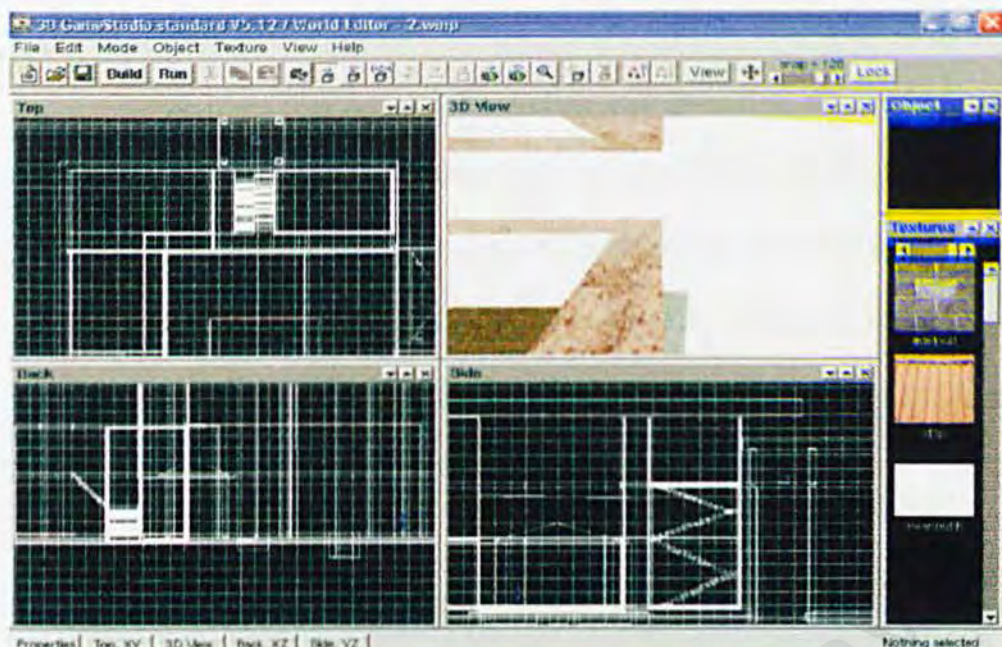


Figure 8.2: Model Being Added to World Design

After all the entities and object have been placed, the world must be compiled and being published.

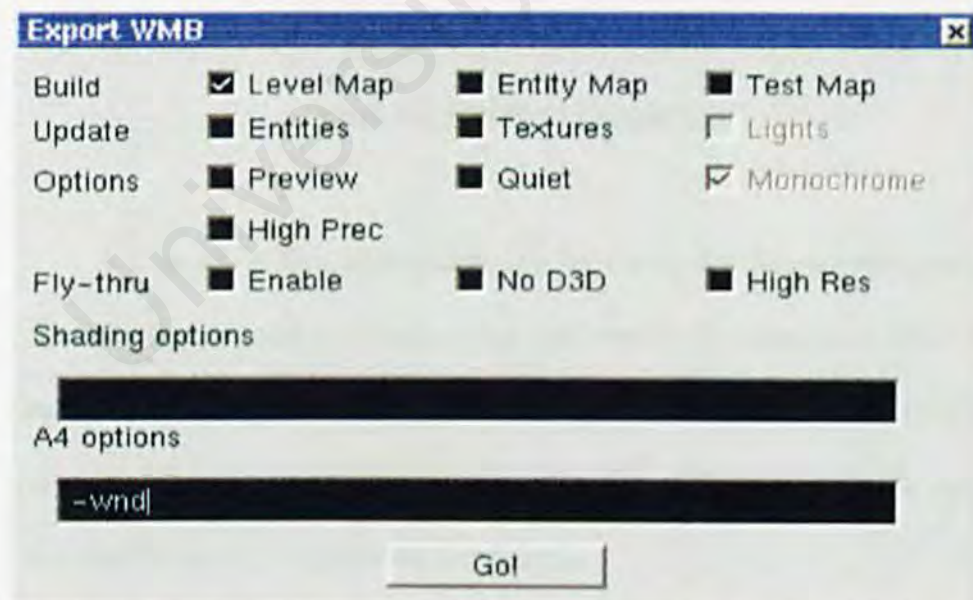


Figure 8.3: Exporting World to Complete Map

8.2.1 Problem Encountered

When I had finished all the design and compiled the world, I had encountered major problem there is my computer could not compile all my work due to lack of RAM on my PC (although 512MB RAM). This is because I had exceeded limit of portal or block in my design. So to overcome the problem I had divided my faculty into 2 section; old and new buildings. After finish all the work again, then I recompiled.

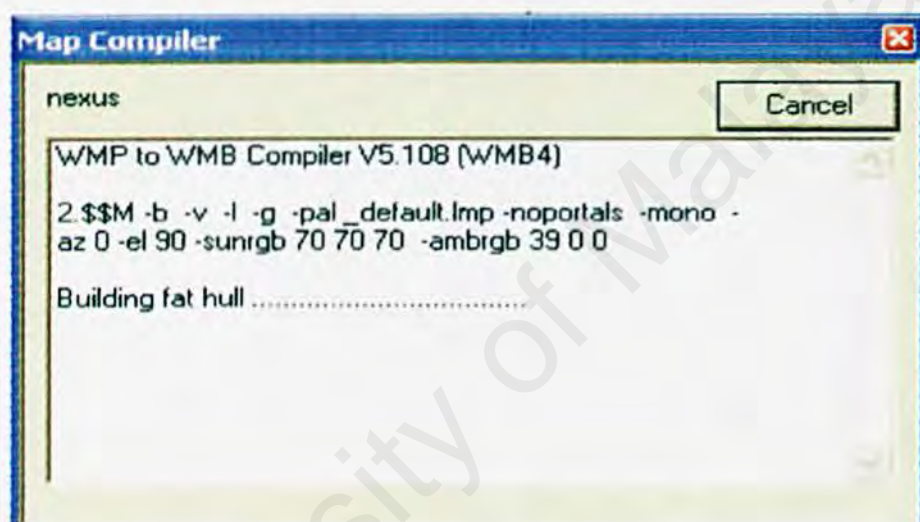


Figure 8.4: Problems Encountered

I hope all of this information can be use in the future enhancement. Actually the systems still have a bug that need to be repaired. I hope this system can be upgrade to a complete faculty, ass well as the design. I will try my best to make this system more effective and efficient in the future because the system is quite new in Malaysia.

8.3 Suggestion

There is lots of suggestion for the future developments:

- i. Make sure have a good PC for developing stage.
- ii. More research have to make for develop a better system.
- iii. Make discussion group (GIS – Group Of Interests) for learning more quickly on software development tools.

CONCLUSION

Conclusion

In conclusion, this system has fulfilled its objectives and requirements. The aim of this project is to develop information system of faculty map through 3D. Some multimedia elements added to the system to make the system not bored users.

This project is very important and beneficial. A lot of knowledge, skills and experience were gained throughout the development of the system. Besides, experience in graphic editing using Adobe Photoshop, 3Dstudio Max, 3DGame Studio and many others give me great experience of how the multimedia elements can be design and give dazzling effects to users.

The most important thing is, I have learned a lot how to find out the solution whenever I encountered problems about developing applications.

Finally, this project has given me a profound impact in time management and communication skills. All the problems faced and experience gained during the system development would be useful in my future career since era is now moving toward 3Ds design and Internet technology that requires decent technical and practical knowledge in development of 3Ds design.

APPENDIX A : USER MANUAL

Chapter 1 - INTRODUCTION

Interactive 3D Tour Around FSKTM is mainly consists of informative map of FSKTM, University of Malaya. This system includes two main building; old building and new building.

This system allow general users and visitors to use the system as a virtual map as its provide information of certain room in FSKTM.

The system is very easy to learn and use. There is a help section being provided, either through the system or using user manual.

This user manual provides the instruction on how to use this system.

1.1 Hardware Requirement

Personal computer with: -

- 64 MB of RAM
- 100 MB of free space in hard disk
- Speaker
- 256-colour monitor capable of resolution 800*600 pixels
- Compatible 3D graphic card (8 MB minimum)

1.2 Software Requirements

Due to the system is stand-alone system; it meets all of the OS (Operating System)

Chapter 2 - GETTING STARTED

This system being provided with a installation files, where the user could install to any directory as they likes, as long as they know where they have put the system folder is.

Figure 2.1 illustrated the main page of the system installation.

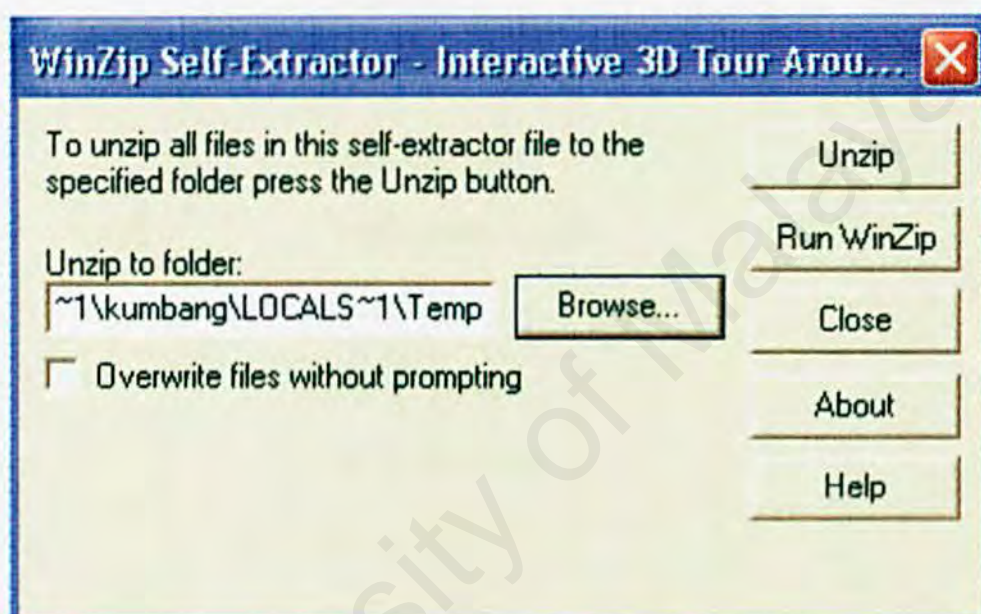


Figure 2.1 – Installation system

User has to set their own directories to install all the requirement files. The default location of installation is in temporary files. User can change to the directories that they like. After user has set the directories they just have to click **UNZIP** and all the installation process have finished.

Chapter 3 – KEY CONFIGURATION

The key configurations are listed below: -

ESC	- Menu
F1	- Help
F6	- Screenshot
F7	- Change Camera
F11	- Toggle gamma
F12	- Toggle sound/music
F10	- Exit
Key Up	- Move Forward
Key Down	- Move Backward
Key Left	- Move Left
Key Right	- Move Right
Key Home	- Jump Obstacle
Key ,	- Look Left
Key .	- Look Right
Key Spacebar	- Action -Open Door etc-
Key M	- Maps
Mouse (Right Click-Hold)	- Run Forward

Users also can lookup the help key in the system by pressing F1 key to activate key help menu.

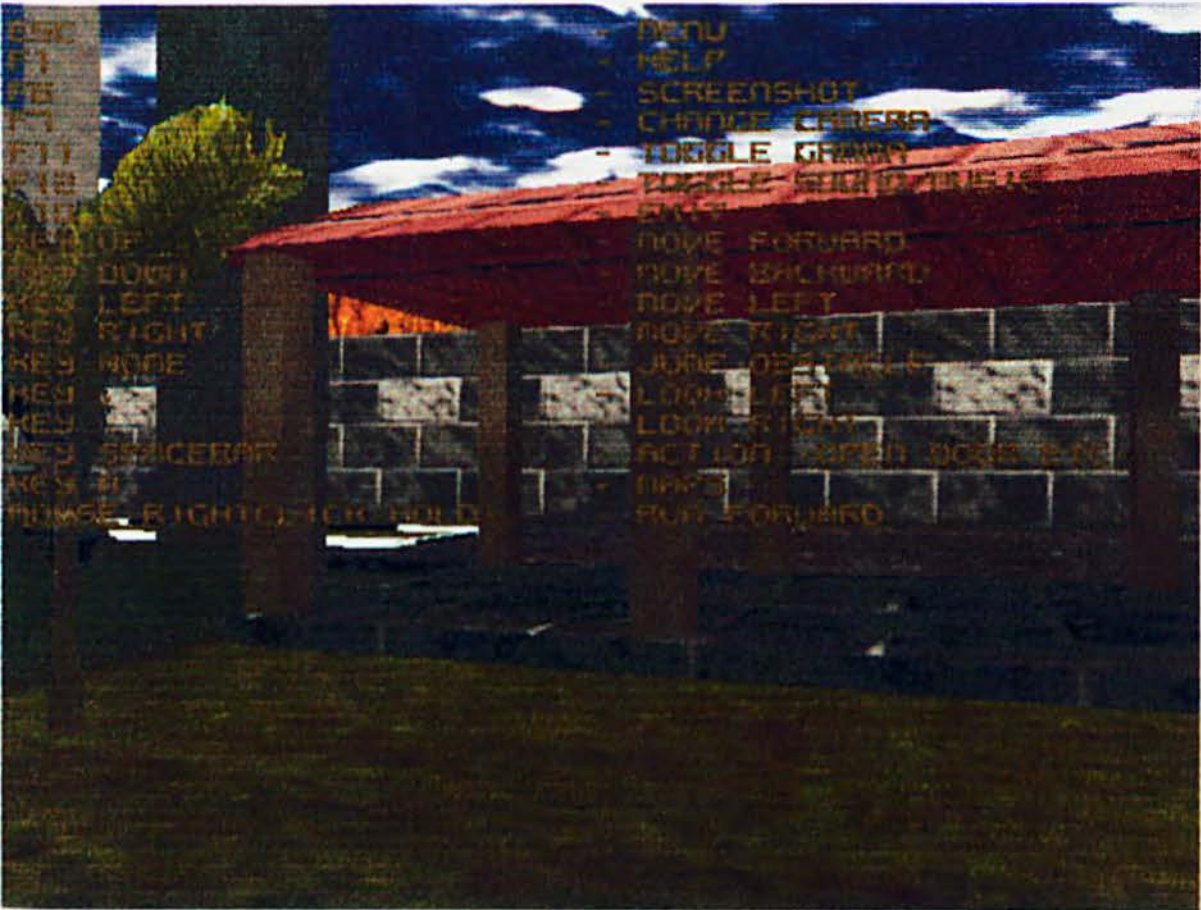


Figure 3.1 – Screen Shot of help key

From now on, user could use the system freely and could interact with all the entities in the system.

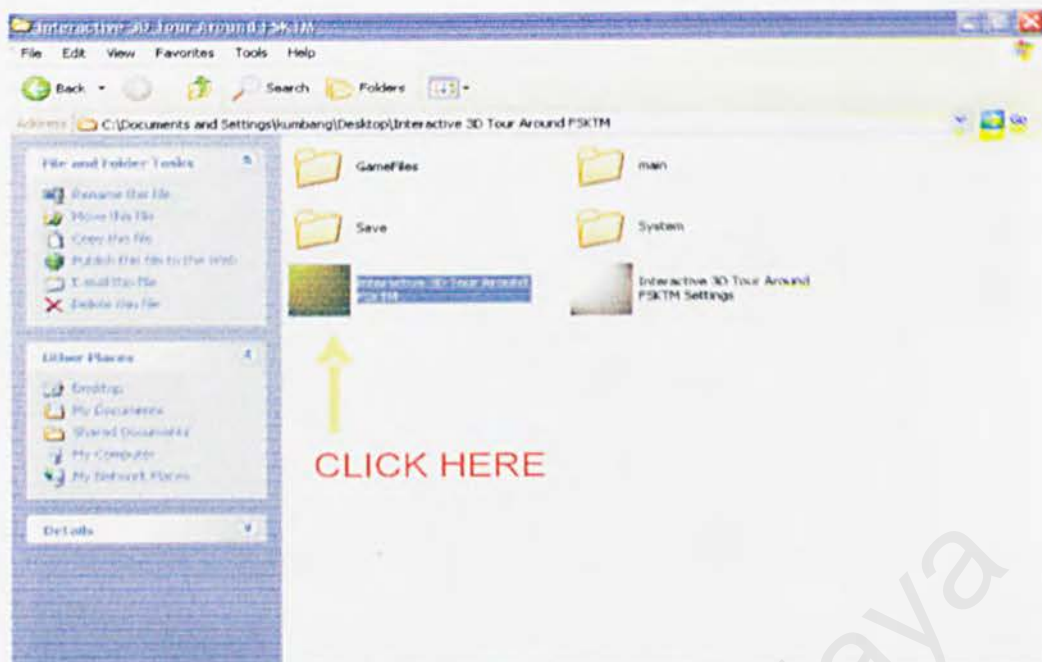


Figure 3.2 – Place where user have to click to start the system



Figure 3.3 – Engine Compiling

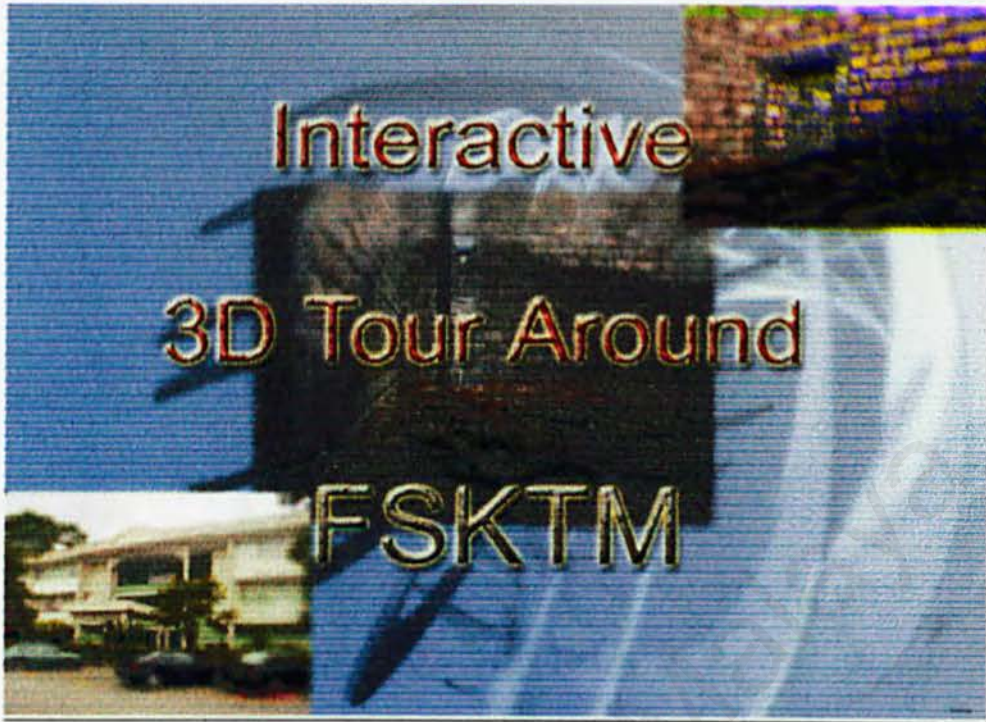


Figure 3.4 – Introduction (Splash Screen)

APPENDIX B : QUESTIONNAIRE

Questionnaires

1. Do you know about 3D?

☐ Yes ☐ No

2. What applications that you know are using 3D design method?

☐ 3D Studio Max ☐ VRML ☐ 3D Games
☐ Maya ☐ AutoCAD ☐ Else (state)

3. Did you know how to design 3D world?

☐ Yes ☐ No

4. What level of your designing in 3D?

☐ Beginner ☐ Advance ☐ Expert

5. Is there any 3D virtual fly-through that you know?

☐ Yes (state) ☐ No

6. Do you think 3D design could be implements in an interactive way? Why?

☐ Yes

☐ No

7. Do you like to view any design in 3D? Why?

☐ Yes

☐ No

8. Do you agree to have 3D-Virtual faculty, as faculty's informational? Why?

☐ Yes

☐ No

9. Would you like to learn how to build "realism 3D's"?

☐ Yes

☐ No

APPENDIX C :

WDLMAN (script)

The Script

```
//1.wdl//
```

```
path "E:\\PROGRAM FILES\\GSTUDIO\\template";
```

```
//////////define//////////
```

```
DEFINE MSG_DEFS;
```

```
DEFINE MSG_X,4; // from left
```

```
DEFINE MSG_Y,4; // from above
```

```
DEFINE BLINK_TICKS,6;
```

```
DEFINE MSG_TICKS,64;
```

```
DEFINE PANEL_POSX,4; // from left
```

```
DEFINE PANEL_POSY,-20; // from below
```

```
FONT digit_font,<digfont.pcx>,12,16;
```

```
FONT standard_font,<ackfont.pcx>,6,9;
```

```
FONT msg_font,<msgfont.pcx>,12,16;
```

```
SOUND msg_sound,<msg.wav>;
```

```
//////////
```

```
//////////include_file//////////
```

```
include <movement.wdl>;
```

```
include <messages.wdl>;
```

```
include <menu.wdl>;
```

```
include <particle.wdl>;
```

```
include <doors.wdl>;
```

```

include <weapons.wdl>;

include <war.wdl>;

include <lflare.wdl>;

include <pintukum.wdl>;

include <kunciku.wdl>;

include <AUDI.wdl>;

include <flash.wdl>;

////////////////////////////////////

var video_mode = 6; // screen size 640x480

var video_depth = 16; // 16 bit colour D3D mode

////////////////////////////////////

string mission_str = "WELCOME TO FSKTM..Press F1 for help";

string level_str = <1.WMB>;

////////////////////////////////////

bmap splashmap = <intro.bmp>;

panel splashscreen

{

    bmap = splashmap;

    flags = refresh,d3d;

}

////////////////////////////////////sky_movement////////////////////////////////////

bmap horizon_map = <horizon.pcx>;

function init_environment()

```



```

{
    scene_map = horizon_map;

    scene_nofilter = on;

    sky_speed.x = 1;

    sky_speed.y = 1.5;

    cloud_speed.x = 3;

    cloud_speed.y = 4.5;

    sky_scale = 0.5;

    sky_curve = 1;

    scene_field = 60;

    scene_angle.tilt = -10;

    sky_clip = scene_angle.tilt;
}

////////////////////main_function////////////////////////////////////

function main()
{
    tex_share = on;      // map entities share their textures

    splashscreen.pos_x = (screen_size.x - bmap_width(splashmap))/2;

    splashscreen.pos_y = (screen_size.y - bmap_height(splashmap))/2;

    splashscreen.visible = on;

    wait(3);

    init_environment();

    level_load(level_str);
}

```

```

}

Panel test_panel2

{
    Layer 2;

    bmap = test_for_5sec2;

    flags = refresh, d3d,transparent;
}

```

```

Panel test_panel3

{
    Layer 2;

    bmap = test_for_5sec3;

    flags = refresh, d3d,transparent;
}

```

```

function papar()
{
    position = camera.z;
    if(position <= -53)
    {
        set test_panel.visible,on;

        wait(80);

        set test_panel.visible,off;

        wait(1);
    }
}

```



```
if(position > 78)
```

```
{
```

```
    set test_panel2.visible,on;
```

```
    wait(80);
```

```
    set test_panel2.visible,off;
```

```
    wait(1);
```

```
}
```

```
if(position <= 78 && position > -53) {
```

```
    set test_panel3.visible,on;
```

```
    wait(80);
```

```
    set test_panel3.visible,off;
```

```
    wait(1);
```

```
}
```

```
}
```

```
on_m = papar();
```

```
//////////window////////////////////////////////////
```

```
WINDOW WINSTART{
```

```
    TITLE            "3D GameStudio";
```

```
    SIZE             480,320;
```

```
    MODE             IMAGE;    //STANDARD;
```

```
    BG_COLOR         RGB(240,240,240);
```

```
    FRAME            FTYPE1,0,0,480,320;
```

```

BUTTON

BUTTON_QUIT,SYS_DEFAULT,"Abort",400,288,72,24;

TEXT_STDOUT      "Arial",RGB(0,0,0),10,10,460,280;

}

////////////////////////////////////

//////////kunciku.wdl//////////

sound key_fetch = <beamer.wav>;

var key1 = 0;

function ambil_kunci(){

    if (indicator != 1) { return; } // must be right kind of scan

    key1 = 1;

    play_sound(key_fetch,50);

    ent_remove(my);

}

action kunci

{

    my.event = ambil_kunci();

    my.enable_scan = on; // pick up by pressing SPACE..

}

function audi()

{

```

```
if (indicator != 1)
```

```
{
```

```
    return;
```

```
}
```

```
if (my._counter <= 0)
```

```
{
```

```
    play_entsound(my,open_snd,66);
```

```
    while (my._counter < 90)        {
```

```
        my.pan -= 3*time; // rotate clockwise
```

```
        my._counter += 3*time;
```

```
        wait(1);
```

```
    }
```

```
    my.pan += my._counter-90; // correct the overshoot
```

```
    my._counter = 90;
```

```
}
```

```
else
```

```
{
```

```
    play_entsound(my,close_snd,66);
```

```
    while (my._counter > 0)        {
```

```
        my.pan += 3*time; // rotate counterclockwise
```

```
        my._counter -= 3*time;
```

```
        wait(1);
```



```

    }

    my.pan += my._counter; // correct the overshoot

    my._counter = 0;

}

}

////////////////////////////////////

```

```

//////////pintukum.wdl//////////

```

```

var indicator = 0;

var my_pos[3];

var my_angle[3];

function scan_me(){

    my_pos.x = camera.x;

    my_pos.y = camera.y;

    my_pos.z = camera.z;

    my_angle.pan = camera.pan;

    my_angle.tilt = camera.tilt;

    temp.pan = 120;

    temp.tilt = 180;

    temp.z = 100; // scan range – 200 quants

    indicator = 1; // this is for opening

    scan(my_pos,my_angle,temp);

}

```

```

define _counter skill25; // use a meaningful name for SKILL25

function door_event(){
    if (indicator != 1)    { return; }

    if (my._counter <= 0) {
        play_entsound(my,open_snd,66);

        while (my._counter < 90) {
            my.pan -= 3*time; // rotate clockwise

            my._counter += 3*time;

            wait(1);
        }

        my.pan += my._counter-90; // correct the overshoot

        my._counter = 90;
    }

    else
    {
        play_entsound(my,close_snd,66);

        while (my._counter > 0)    {
            my.pan += 3*time; // rotate counterclockwise

            my._counter -= 3*time;

            wait(1);
        }

        my.pan += my._counter; // correct the overshoot

        my._counter = 0;
    }
}

```

```

    }
}

action pintuku{

    my.event = door_event;

    my.enable_scan = on; // make door sensitive for scanning

    my._counter = 0;

}

```

```

action pintuku2{

    my.event = door_event;

    my.enable_scan = on; // make door sensitive for scanning

    my._counter = 0;

}

```

```

//////////

```

```

string B2_str = "Makmal Kejuruteraan Perisian..

```

```

You are not allowed to enter this room !";

```

```

text B2_did{

    pos_x = 300;

    pos_y = 240;

    font = msg_font;

    string = B2_str;

    layer = 10;

```



```

        flags = transparent, d3d, center_x, center_y, narrow;
    }

function text_B2(){

    set B2_did.visible,on;

    waitt(80);

    set B2_did.visible,off;

    wait(1);

}


action pintu_B2{

    my.event = text_B2;

    my.enable_scan = on; // make door sensitive for scanning

    my._counter = 0;

}

////////////////////////////////////

////////////////////////////////////

string B3A_str = "Lecture Room 1A..

You are not allowed to enter this room !";

text B3A_did{

    pos_x = 300;

    pos_y = 240;

    font = msg_font;

```

```

    string = B3A_str;

    layer = 10;

    flags = transparent, d3d, center_x, center_y, narrow;

}

function text_B3A(){

    set B3A_did.visible,on;

    waitt(80);

    set B3A_did.visible,off;

    wait(1);

}

action pintu_B3A{

    my.event = text_B3A;

    my.enable_scan = on; // make door sensitive for scanning

    my._counter = 0;

}

////////////////////////////////////

////////////////////////////////////

string B3B_str = "Lecture Room 1B..

You are not allowed to enter this room !";

text B3B_did{

    pos_x = 300;

    pos_y = 240;

```

```

font = msg_font;

string = B3B_str;

layer = 10;

flags = transparent, d3d, center_x, center_y, narrow;

}

function text_B3B(){

    set B3B_did.visible,on;

    waitt(80);

    set B3B_did.visible,off;

    wait(1);

}

action pintu_B3B

{

    my.event = text_B3B;

    my.enable_scan = on; // make door sensitive for scanning

    my._counter = 0;

}

////////////////////////////////////

////////////////////////////////////

string B5_str = "Canteen.. You are not allowed to enter this room !";

text B5_did{

    pos_x = 300;

```



```

pos_y = 240;

font = msg_font;

string = B5_str;

layer = 10;

flags = transparent, d3d, center_x, center_y, narrow;

}

function text_B5(){

    set B5_did.visible,on;

    waitt(80);

    set B5_did.visible,off;

    wait(1);

}

action pintu_B5

{

    my.event = text_B2;

    my.enable_scan = on; // make door sensitive for scanning

    my._counter = 0;

}

////////////////////script for dynamic light////////////////////

VAR Light_on;

ON_E set_light;

var MyPos[3];

```

```

FUNCTION set_light (){
    IF (Light_on == 0)    {
        Light_on = 1;
        CREATE <trigger.wmb>,PLAYER.POS,light_akt;
    }
    ELSE { Light_on = 0;}
}

```

```

FUNCTION light_akt (){
    MY.INVISIBLE = ON;
    MY.PASSABLE = ON;
    WHILE (Light_on == 1)    {
        MyPos.X = 5000;
        MyPos.Y = 0;
        MyPos.Z = 0;
        VECROTATE MyPos,CAMERA.PAN;
        VEC_ADD (MyPos,CAMERA.POS);
        LOOK NULL,CAMERA.POS,MyPos;
        IF (RESULT > 0)
        {
            VEC_SET (MyPos,TARGET);
            temp.X = NORMAL.X * 20;
            temp.Y = NORMAL.Y * 20;
            temp.Z = NORMAL.Z * 20;

```

```

        VEC_ADD (MyPos,temp);

        VEC_SET (MY.POS,MyPos);

    }

    MY.LIGHTRANGE = vec_dist(camera.pos,my.pos) * 0.5;

    MY.LIGHTRED = 200 - ( vec_dist(camera.pos,my.pos) * 0.1 );

    MY.LIGHTGREEN = 200 - ( vec_dist(camera.pos,my.pos) * 0.1 );

    MY.LIGHTBLUE = 200 - ( vec_dist(camera.pos,my.pos) * 0.1 );

    WAIT (1);

}

REMOVE ME;

}

```

```

//////////AUDI//////////

```

```

DEFINE MSG_DEFS;

DEFINE MSG_X,4; // from left

DEFINE MSG_Y,4; // from above

DEFINE BLINK_TICKS,6;

DEFINE MSG_TICKS,64;

DEFINE PANEL_POSX,4; // from left

DEFINE PANEL_POSY,-20; // from below

```



```

/*FONT digit_font,<digfont.pcx>,12,16;

FONT standard_font,<ackfont.pcx>,6,9;

FONT msg_font,<msgfont.pcx>,12,16;*/

SOUND      msg_sound,<msg.wav>;

////////////////////////////////////

////////////////////////////////////string////////////////////////////////////

STRING need_key1_str,"You need a key to open this door !!";

STRING got_key1_str,"FOUND A KEY";

////////////////////////////////////

////////////////////////////////////action////////////////////////////////////

ACTION big_tor_norm {

    MY._PAUSE = 80;

    BRANCH gate;

}

ACTION big_tor {

    MY._PAUSE = 80;

    MY._KEYTYPE = 1;

    BRANCH gate; }

////////////////////////////////////

STRING textap_str = "Approaching a DOOR..need a key first !!";

text you_did{

pos_x = 300;

pos_y = 240;

```

```
font = msg_font;  
string = textap_str;  
layer = 10;  
flags = transparent, d3d, center_x, center_y, narrow;  
}
```

```
//////////new//////////
```

```
function text(){  
    set you_did.visible,on;  
    wait(80);  
    set you_did.visible,off;  
    wait(1);  
}
```

```
ACTION Show_pictext {  
    SET MY.ENABLE_IMPACT, ON;  
    SET MY.EVENT text;  
}
```

REFERENCE

PROJECT
SCHEDULE

University of Malaya

APPENDIX D : PROJECT SCEDULE

Reference:

- 1 Beatty, John C., and Kellogg S. Booth, "Tutorial: Computer Graphics", 2nd.ed. IEEE Computer Society, 1982.
- 2 Salmon, Rod and Mel Slatter, Computer Graphic Systems & Concepts, Addison – Wesley Publishing Company, 1987.
- 3 John Vince: Essential Computer Animation Fast. London: Springer, 2000.
- 4 Newman, William M., and Robert F. Sproull : Principles of Interactive Computer Graphics, 2nd.ed., Tokyo : McGraw-Hill Kogakushan Ltd., 1979.
- 5 John Vince. Computer Graphics for Graphics Designers. White Plains, New York: Knowledge Industry Publications, Inc., 1985.
- 6 Donald Hearn, and M. Pauline Baker, Computer Graphics, 2nd.ed., Prentice Hall, 1994.
- 7 R.R Everett: "The Whirlwind 1 Computer," Joint AIEE-IEE Conf., 1952, Rev. electron, digital Comput., February 1952, pp.70. (Extracted by Newman and Sproull)
- 8 HowStuffWork, Inc. 2003 *Internet user's Guide* [online]. USA. Available from <http://www.howstuffworks.com>. [Accessed 15 May 2003]
- 9 A CONNECTPRESS Ltd. Publication [online]. USA. Available from http://www.proe.com/Summary/sum_mt02S.htm. [Accessed 18 May 2003]
- 10 MROZ, 2003 *Homepage of Lucas MROZ* [online]. Austria: Vienna University of Technology. Available from

<http://www.cg.tuwien.ac.at/~mroz/sirds/history.html>. [Accessed 18 May 2003].

- 11 A.D Brown, 2001 *Leeds University as a virtual university* [online]. UK. Available from <http://www.misconception.org.uk/andy/project/report/node45.html> [Access on 15 May 2003].
- 12 Whitten, P.H, Bentley, L.D., Dittman K.C. (2000). Requirement discovery. In Whitten, P.H. System Analysis and Design Methods (5th ed.). New York: McGraw-Hill.
- 13 Conitec Corporation., 2003 *3DGameStudio* [online]. USA. Available from <http://www.conitec.net/a4info.htm> [Access on 20 May 2003]